PRIJS ƒ 7.95/Bfr. 160

Commodore INFO

ONAFHANKELIJK BLAD VOOR COMMODORE GEBRUIKERS

JAARGANG 6, NO. 4, juni/juli 1989



LISTINGS

- Omzetter C-16
- Hardcopy C-16
- Dropper C-16
- Golf C-64
- Key Maze C-64
- Krantkolom C-64
- Ruimtelanding C-64
- Sup.write C-64
- Darts C-64
- Woordtraining C-64
- Rubik's Clock Amiga

Audiomaster II Amiga

Rendale Genlocker

Capone

Draw 2000

1581 diskdrive

Captain Fizz

Tips & Trucs

Vaste rubrieken

Geos Info

Basic Miniatuurtjes

AmigaDOS cursus

C-64 Machinetaal

Commodore Info

Verschijnt 8x per jaar
Jaarg.6, no.4, juni/juli 1989

Uitgave:
Sala Communications
Uitgever:
Vic Sharfman
Redactie:
Ir. L. Sala hoofdredacteur
J. Bodzinga adj. hoofdred.
drs. J. Boers eindredacteur
drs. M. de Rooij &
J. Broekhuizen productie
drs. H. Zoete, H. Smeenk, drs. U. Schuurmans, R. Goudriaan, B. Munniksma, B. Venema, P. Boncz, MGCC/Johan & Johan

Redactiesecretariaat: R. van Zalingen
Strip: Bert Tier
Illustraties: Ben van Mierlo

Advertentie-exploitatie:
Ing. V. Sala, Ing. B. Sala,
D. van Vlijmen
Weesperstraat 103
1018 VN Amsterdam
tel. 020-273198
Redactie adres:
Postbus 43048
1009 ZA Amsterdam
tel. 020-228871
Listingtelefoon:
(ma: 17.00-21.00) 02155-25162

Abonnementen en administratie:
Nicole Balke en Marjo Jansen
Postbus 43048
1009 ZA Amsterdam
tel. 020-248006
Vragen betreffende abonnementen ontvangen wij bij voorkeur schriftelijk, met meesturen van het omslagetiket.
Abonnement:
Voor 8 nummers ƒ 47,50 of Bfr. 975 per jaar. Betaling op giro 1585491 (België: BBL nr. 310050602562) t.n.v. SAC/Commodore-Info. Oude nummers kunt U alleen krijgen bij vooruitbetaling van ƒ 6,75 op de bovenstaande rekening. Ook telefonische opgave voor een abonnement is mogelijk. Bel GRATIS 06-02242222 (teleservice), elke dag tot 20.20 uur (dus ook in het weekend). België: 115555, dagelijks tot 22.00 uur. Deze telefoonnummers zijn alleen bedoeld voor opgave van NIEUWE abonnementen.
Opzegging dient schriftelijk te geschieden uiterlijk twee maanden voor de aanvang van een nieuwe abonnementsperiode van een jaar.

Omslagfoto: Audiomaster II (Aegis)

Zetwerk & druk: NDB, Zoeterwoude
Distributie:
In Nederland: Betapress, Gilze
In België: AMP, Brussel

ISSN: 0169-3085

Inhoud van dit nummer

Redactioneel

Commodore is op de Amerikaanse markt lang niet meer zo belangrijk als in het verleden. Qua omzet moet men het vooral van Duitsland hebben, maar toch komt de meeste software wel uit de VS, met Engeland als goede tweede. Een klassiek patroon, al moeten we wel toegeven dat het merendeel van de titels eerst voor PC uitkomt, maar de Amiga versie is meestal fraaier. Rondkijkend op een aantal beurzen in de VS viel me weer op, dat er langzamerhand toch wel heel goede Amiga software is, ook voor serieus werk.
De tekstverwerker ProWrite 2.0 van New Horizons bijvoorbeeld doet is een goed, goedkoop, en niet te moeilijk pakket, werkt in en met kleur en graphics (DTP!), en heeft een goede ondersteuning in de vorm van font-pakketten en een echte PostScript utility ProScript. WordPerfect voor de Amiga is nog beter, maar relatief duur.
Games, games, games, wie op zo'n beurs de tijd wil nemen kan zijn hart ophalen. Amiga spellen worden hele films op zich, zoals bijvoorbeeld Sword of Sodan van Discovery Software, dat maar liefst 4 MegaBytes beslaat. Dragon's Lair van Readysoft is nog veel groter. De flight simulators worden steeds mooier, programma's als Falcon (SH), Interceptor, Starglider II en Hawk (EA) zijn spectaculair.
De A-MAX Mac emulator van Readysoft belooft de software-rijkdom van de Apple Mac's op de Amiga, maar we kennen nog geen importeur van dit produkt. Bovendien zijn er (echte) Apple ROM's nodig om ermee te werken en een Mac floppy drive.
Het lijdt geen twijfel, de Amiga leeft, maar daarbij zien we helaas voor de C-64 bezitter steeds minder nieuws. Robocop van DataEast is een uitzondering, meestal mag de 64 bezitter pas aansluiten, als er een spel het zo goed doet, dat men er ook een 64 versie van maakt. Jammer, maar toch blijft Commodore Info de 64 en 128 gebruikers trouw!

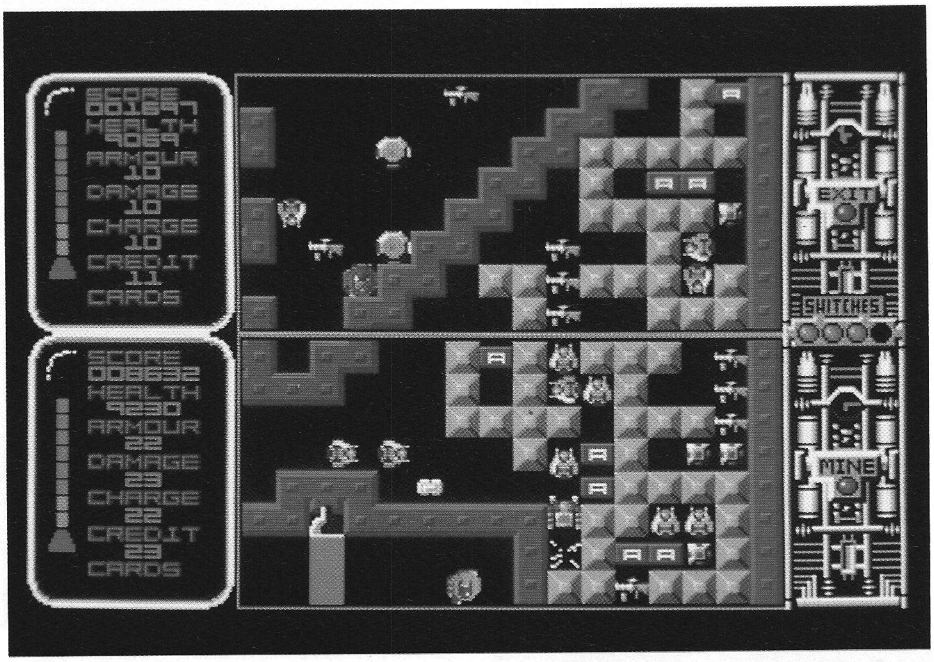
- Luc Sala

Het spelen van spelletjes op de computer is zo oud als de uitvinding van deze machine. Maar om samen met iemand anders een boeiende middag of avond door te brengen is een heel ander verhaal. Slechts enkele games zijn van echt goed niveau . Gelukkig is er aan het kleine lijstje teamspellen weer een nieuwe naam toegevoegd.

Captain Fizz

Enige tijd geleden zei een vriend tegen mij: 'Goh, op zichzelf is het spelen van spelletjes wel leuk, maar over het algemeen speel je ze meestal alleen.' En na enige zoekwerk moest ik hem wel gelijk geven. Op het gebied van teamspellen waren er niet meer dan vijf interessante spellen te vinden. Psyclapse heeft er hier sinds kort een aan toegevoegd.

Het spel Captain Fizz. Dit spel, dat zowel voor de 8-bits als voor de 16-bits computers te verkrijgen is, is gek genoeg niet een van Psyclapse's grafische hoogtepunten te noemen, als je dit spel met hun voorgaande successen vergelijkt. Hij is in vergelijking tussen Amiga en 64 echter verrassend goed afgewerkt. Natuurlijk gaat het op de Amiga soepeler en is het geluid zonder meer superieur, maar afgezien van deze feiten, is hij op de 64 perfect speelbaar. Het verhaal speelt zich af op een ruimteschip dat zich, door een groot aantal vijandelijke individuen op weg naar de zon begeeft. Om het schip en passagiers te redden zijn er twee Huurlingen aan boord van het ruimteschip geteleporteerd en moeten zij als een TEAM de brug van het ruimteschip bereiken.



Schermbeeld van Captain Fizz.

Energiedeuren

Op hun weg komen zij zowel vijanden als opstakels tegen. Deze vijanden variëren van gewoon kanonnenvoer tot zwaar bewapende tanks. De obstakels zijn normale, energie-deuren en barricades. Voor de gewone deuren zijn pasjes nodig. er zijn op elk level vier kleuren te vinden en voor elke kleur deur zijn er pasjes. De energiedeuren zijn een ander probleem. Deze moeten door een combinatie van vier knoppen geopend worden. Deze deuren sluiten zich na enige seconden weer, dus is het zaak dat de deur zo snel mogelijk gepasseerd wordt. Dit is vaak alleen mogelijk zo'n deur te passeren door speler 1 bij de deur te laten staan, speler 2 de deur te laten openen en speler 1 de deur laten passeren. Daarna vanaf de andere zijde speler 2 hetzelfde laten doen zodat ook speler 1 verder kan. De bedoeling is om als een TEAM door de diverse problemen te komen. Indien je dit achterwege laat kom je er op sterfelijke wijze achter, dat in een goed kruisvuur een goed schutter geliefder is dan een hogere score. Ik kan niet anders zeggen dan dat het spel zonder meer garant staat voor meerdere dagen joystickplezier. Psyclapse bewijst tevens dat het spel de gelukkige speler maakt en niet de hi-res beeldschermen.

Het spel is voor *f* 50,- te krijgen bij de betere spellen leverancier.
Het programma is verkrijgbaar voor de C-64 en Amiga, en daarnaast nog voor de Spectrum, PC en de Atari ST.

GRAFISCH	60
GELUID	60
SPEL	95
TOTAAL	71

Er zijn in Nederland een aantal leveranciers die het steeds weer blijven proberen om software te importeren. Het is niet eenvoudig, elke keer als een spel op de markt komt (en vaak daarvoor äl)wordt het meteen al gekopieerd en in het illegale circuit rondgedeeld.

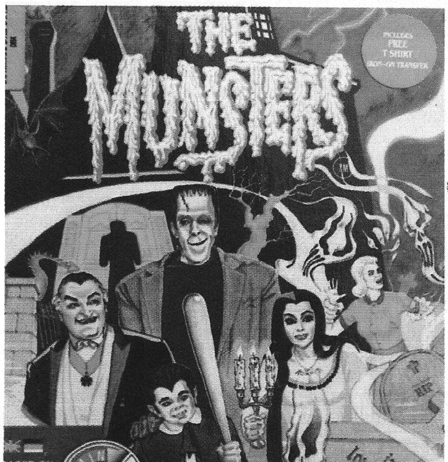
Krachtpatsers, spoken en balkunstenaars

Er gaat een gerucht dat van elk spel dat wordt verkocht er 25 kopieën worden gemaakt. U kunt nagaan hoe moeilijk het is om toch, voor de grote groep die hier niet aan mee willen doen, software op de markt te brengen.

HomeSoft Benelux is zo'n bedrijf dat ondanks dit alles het toch blijft proberen om tegen een redelijke prijs software te leveren. Zij stelden ons een groot aantal pakketten ter hand om hier wat over te schrijven. Dit aantal is zo groot en er zijn zulke uitgebreide spelen bij dat we dit niet in één keer kunnen redden. Er blijft dus nog wat over voor het volgend nummer.

De Monsters

Deze software is van een minder bekend Amerikaans Softwarehuis, Harribly Good Software. Minder bekend slaat hierbij zeker niet op minder goed. De monsters is een ras echt arcade spel, dus vol aktie. De opdracht is een monster familie te gaan helpen. Marilyn is een de kwade greep van Old Nick gekomen. Jij moet nu proberen haar uit zijn klauwen te gaan redden. Om dit te kunnen bereiken moet



The Munsters



Dragonninja

je van je tover krachten gebruik gaan maken. Dit is de enige manier om de kwade geesten terug te sturen naar de onderwereld. Al deze handelingen kosten veel energie, dus spring hier voorzichtig mee om. Er zijn een groot aantal mysterieuze objecten in het parcours verstopt. Probeer deze allemaal op te sporen en verzamel ze. Dit moet in de goede volgorde gebeuren, dus pas op met wat je doet. Zombies, geesten en vampiers, deze kun je op je weg tegen komen, zij maken het de familie Monster (en dus ook jou) steeds weer moeilijk. Deze arme familie weet niet wat ze overkomt, altijd zijn ze aardig, en misschien daarom gaat het steeds weer fout, ze worden steeds weer het slachtoffer van de onderwereldse vandalen. Je hebt niets anders nodig dan je joystick, geluk, verstand en heel veel geduld om je opdracht tot een goed einde te brengen. Veel oefenen en zorgen dat je nachtrust er niet van te leiden heeft, dan kan dit spel je veel plezier verschaffen. Het spel is verkrijgbaar voor de Commodore 64 en de Atari.

Dragonninja

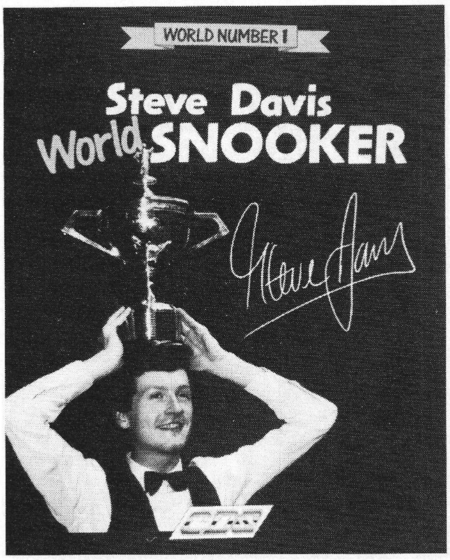
Data East is de ontwerper van dit aktiespel. Het is een spel, waar hardheid en vechtlust op de eerste plaats komt. Echt dus een spel voor de liefhebbers van de karate sport. Het zijn maar liefst zeven levels, waarbij slaan, ontwijken, springen en niet te vergeten schoppen, een eerste vereiste zijn om te overleven. Het vergt

veel oefening om de held die beweging te laten maken die je wilt. Onze held wordt bestuurd met de joystick. We kunnen de joystick in acht richtingen sturen, deze corresponderen met acht verschillende . funkties. Maar drukken we nu gelijktijdig de vuurknop in dan staan ons nog acht extra funkties ter beschikking. In het totaal kunnen we onze hoofdfiguur zestien verschillende bewegingen laten maken. In de bijgeleverde handleiding, helaas niet in het Nederlands, staan de verschillende technieken beschreven. Aan het eind van ieder level, als het gevecht is gewonnen, wacht je een speciale opdracht. Een vuurspuwende dikke boosaardige man, probeert je de weg te versperren. Deze groene gigant Ninja heeft er de gewoonte van gemaakt om plotseling te verschijnen. Heb je je opdracht volbracht, juich dan dus niet al te vroeg. Zo tussendoor moet eerst de president van de Verenigde Staten worden gered. Lukt dit je dan vlieg je samen met hem naar veilige oorden. Maar voor dat dit lukt zal er nog wel wat water door de Rijn stromen. Kortom, een spel voor de liefhebber. Verkrijgbaar voor de Commodore 64

Snooker

Als we aan Snooker denken, dan denken we natuurlijk aan Engeland. Daar is één man die dat spel op een voortreffelijke manier beheerst, dan hebben we het natuurlijk over Steve Davis. Vorige maand weer Wereld kampioen Snooker geworden. Bezit U nu een Amiga, dan is het aanschaffen van dit spel goed voor urenlang gekluisterd te zitten aan de computer.
World Snooker, zoals dit spel eigenlijk heet, bestaat uit vijf verschillende spelen. Ieder met zijn eigen (on)mogelijkheden. Snooker, Pool met Engelse regels, Pool met Amerikaanse regels, Engels billiard en de minder bekende Carom Billiards. Elk spel wordt volgens de normale spelregels gespeeld.

Snooker (Amerikaans) wordt gespeeld met twee spelers en een biljard met een aantal opvang zakken, in elke hoek en twee in het midden van de lange zijden. 15 rode ballen, en zes gekleurde ballen en een witte speelbal. De speler met de meeste punten is de winnaar. De ballen zijn verschillend van waarde. Rode ballen hebben de laagste waarde, 1 punt. De gele bal 2 punten, de groene 3 punten, de bruine 4, blauw 5, rose 6 en de zwarte is het duurste deze is 7 punten waard. We beginnen het spel waarbij



World Snooker

de gekleurde ballen in formatie zijn opgesteld en onze witte speelbal moet ergens in het "D" vlak liggen. We kunnen zelf bepalen waar we, binnen deze grenzen, beginnen. We moeten altijd een rode bal raken, anders gaan er punten naar de tegenpartij. Deze punten gaan ook naar je tegenstander als ongelukkiger wijs je speelbal in één van de zakken verdwijnt. Hierna start de tegenstander weer uit de begin positie, dus uit de "D". Heb je een rode bal niet alleen geraakt, maar ook in de zak gespeeld dan mag je een andere kleur bal proberen in één van de zakken te laten verdwijnen. Dus hiermee zijn de meeste punten te verdienen. De gekleurde ballen worden nadat ze in een zak verdwenen zijn weer terug gelegd op de tafel, op hun eigen vastgestelde plaats. Hierna moet altijd weer eerst een rode bal gespeeld worden. Rode ballen worden niet terug gelegd, anders zou het spel nooit ten einde komen. Het is dus zaak niet alleen te scoren, maar ook de ballen zo moeilijk mogelijk voor je tegenstander te laten liggen.
Het Engelse spel wordt gespeeld met een speelbal en 15 andere kleuren ballen. Eén zwarte bal en twee sets van zeven ballen, rood en geel. De bedoeling is alle ballen van je eigen kleur + de zwarte in de zakken te spelen voor je tegenstander het zelfde met zijn ballen heeft gedaan. Wie dat het eerste voor elkaar heeft heeft dit spel gewonnen.
USA pool wordt gespeeld op een hard blauw laken, met een witte bal en 15 genummerde ballen. De eerste bal mag op elke positie achter de zwarte lijn worden neergelegd om te worden gespeeld. Er moet altijd eerst een bal worden aangewezen, welke men wil gaan spelen, daarna moet worden aangegeven in welke zak hij wordt gespeeld. Als alles wordt gemist gaan de punten weer naar de tegenstander. Uitkijken is dus een eerste vereiste.
Engels biljard wordt gespeeld met twee witte ballen en een rode bal. Er moet bij het begin worden aangegeven tot hoeveel punten men het spel wil gaan spelen. Er zijn drie manieren om punten te verdienen, scoren, missen van de tegenstander en het maken van een carambole. Punten worden bij elkaar opgeteld door het maken van een carambole en de bal verdwijnt in één van de zakken levert de meeste punten op. Ook hier mag worden af begonnen op elk punt binnen het afgeschermde "D" veld. Bij deze eerste bal ligt de bal van de tegenstander nog niet op tafel.
Carom biljards lijkt het meest op ons biljard wat er overal in Nederland en Belgie wordt gespeeld. Het spel wordt gespeeld op een biljard zonder zakken. We hebben op de tafel 1 rode bal en twee witte ballen, waarvan er één een stipje heeft. Iedere keer dat en rode en de witte bal van de tegenstander wordt geraakt levert 1 punt op. Lukt dit niet dan is de tegenstander aan de beurt. Ook hier moet aan het begin de hoogte van de score worden aangegeven tot waar men het spel wil gaan spelen.
Het spel is volledig menu gestuurd. De snelheid die de bal moet krijgen wordt onder aan het scherm aangegeven door middel van een balk. De richting van de bal wordt aangegeven door de lijn te trekken vanuit de speelbal in de richting waar men de bal heen wil gaan spelen. Tot slot kan ook worden aangegeven waar men de bal wil gaan raken, met andere woorden wat voor effect moet er aan de bal worden gegeven. Zelfs is het mogelijk dat je de Amiga als tegenstander gebruikt. Maar dan is het wel moeilijk winnen, we hebben dus gewaarschuwd. Ook voor de C-64.

Circus games

Programma's met verschillende spelen zijn er al in verschillende soorten en maten uitgebracht. Denk hierbij maar eens aan de legendarische zomer- en winterspelen. TyneSoft heeft samen met HomeSoft aan deze reeks een nieuwe loot toegevoegd. Circus Games. Als liefhebber van dit soort spelen begint dan natuurlijk je hart sneller te kloppen. De verwachtingen zijn hoog gespannen, nog mooier werd het bij het openen van de doos. Ingesloten was een komplete handleiding, ja, in het Nederlands. Dit komen

we bij de software pakketten niet vaak tegen. Dus met volle moed ertegen aan. Door een volleerd spreek stalmeester wordt U welkom geheten door de firma TyneSoft in het grootste circus spektakel aller tijden. Ringling Bros & Barnum brengen U de circus games, iets geheel nieuws op het gebied van spanning en sensatie. Het spel laat U kennis maken met de verschillende onderdelen van het circus. Zo kan U eindelijk eens hoog in de nok aan de trapeze hangen, ja, zelf als U hoogtevrees heeft. Het wankele koord blijkt een grote aantrekkings kracht te hebben, bokkesprongen maken op de paarden, het temmen van leeuwen, allemaal onderdelen waar U al jaren naar uitgekeken heeft, en dit alles vanuit uw rustige zolder kamertje achter de vertrouwde computer. Alle onderdelen kunnen geoefend worden, maar wilt U dat uw naam gaat prijken om de erelijst dan zit er niets anders op dan mee te doen aan de internationale competitie. Een moeilijk onderdeel blijkt het koordansen te zijn. Tijdens het lopen over het slappe koord moeten er een groot aantal sprongen worden uitgevoerd. Een handstand, een salto en een



Circus Games

radslag, allemaal onderdelen die hier moeten worden uitgevoerd. Lukt dit dan komt zo mogelijk het moeilijkste, met een éénwieler balanceren op het koord. Alles hoeft maar één keer te worden uitgevoerd om een zo hoog mogelijk punten aantal te behalen. Hierna worden de paarden ongeduldig. Het paard loopt een aantal rondjes in de piste, waarbij het de bedoeling is om op zijn (of haar) rug te gaan staan. Nu zal blijken dat dit niet meevalt. En alsof het nog niet moeilijk genoeg is, ook hier ontkom je er niet aan, een salto en verschillende draaien, zijn het verplichte onderdeel. Tijdens het trapeze werk houd het publiek zijn adem in, wat zal hij er van terecht brengen. Het trapeze werk is verdeeld in drie onderdelen. Oplopend in moeilijkheids graad, moeten er verschillende draaiingen worden gemaakt tijdens de vlucht. Het laatste onderdeel is misschien niet het moeilijkste maar wel het gevaarlijkste onderdeel. Een stoel en een zweep is de enigste bescherming die je hebt. Sla hier niet te veel mee, je wekt er alleen maar meer agressie mee op bij onze viervoeters. De tijgers moeten door de brandende hoepel springen, dit levert punten op, elke fout kost punten.

Al met al zijn onze verwachtingen niet te hoog gespannen geweest, een leuk, gevarieerd spel, het is weer eens iets anders. Het programma is voor de commodore 64 leverbaar

De tweede aflevering van de cursus Graphics van Hylke Sprangers en Michel de Boer gaat over sprites. Ook deze aflevering zal redelijk simpel beginnen, zodat iedereen het kan begrijpen, en wat moeilijker eindigen, voor de gevorderden. Omdat er over sprites zoveel te vertellen is, besteden we niet een maar twee afleveringen aan dit onderwerp.

GRAPHICS

Deel 2: de sprites

In de vorige aflevering zijn we begonnen met het behandelen van een van de drie grafische objecten die de Commodore kent: de karakters. Dit waren kleine figuurtjes die vooral geschikt waren voor het weergeven van tekst en het maken van achtergronden. Maar om echt flitsende beelden te maken zijn deze objecten niet geschikt. Deze keer gaan we een stap verder: we zullen de sprites gaan behandelen. Dit zijn waarschijnlijk de meest fascinerende grafische objecten die de Commodore 64 rijk is. Dit komt onder andere omdat ze betrekkelijk makkelijk te maken zijn en heel geschikt zijn voor animatie. Kortom, het betere werk is aangebroken.

In het kader van de sprites zullen we de volgende items behandelen:

° wat is een sprite
° opbouw van sprites
° sprites op het scherm zetten
° kleuren
° sprites vergroten
° animatie van sprites
° applicatie programma's

Wat is een sprite?

Een sprite is een beweegbaar figuurtje, in het engels ook wel een mob genoemd (moveble object). Dit figuurtje is ongeveer acht keer zo groot als een karakter. In tegenstelling tot een karakter kan een sprite op elke willekeurige plaats op het scherm worden gezet (een karakter kan immers maar op 1000 plaaten gezet worden). De kleur van een sprite kan gekozen worden uit de 16 standaard kleuren. Standaard kunnen er acht sprites tegelijk op het scherm worden gezet. Elk van deze sprites kunt u zelf definieren. Net als bij de karakters zijn er voor sprites verschillende kleurenmodes. Een groot voordeel van sprites ten opzichte van karakters is dat ze zeer eenvoudig te bewegen zijn.

Opbouw van een sprite

Een sprite bestaat uit 504 pixels. Deze pixels staan in een raster (matrix) van 21 pixels verticaal en 24 pixels horizontaal. Elk van deze 504 pixels kan aan of uit worden gezet. Ook hier wordt een pixel intern gerepresenteerd door een bit. Zoals u al weet, uit de vorige aflevering, gaan er acht bits in een byte (geheugenadres). Acht pixels naast elkaar vormen zo samen een byte. Aangezien een horizontale rij pixels van een sprite uit 24 bits bestaat, kan de hele rij dus gerepresenteerd worden door drie bytes. Om het geheel te verduidelijken staat in figuur 1 de ordening van de bytes in een sprite. Byte 0, byte 1 en byte 2 vormen hier de eerste rij pixels. Uit dit figuur blijkt dat een sprite uit 3 * 21 = 63 bytes is opgebouwd.

Bytevolgorde in sprite

byte 00	byte 01	byte 02
byte 03	byte 04	byte 05
byte 06	byte 07	byte 08
:	:	:
byte 60	byte 61	byte 62

Figuur 1

Om nu zelf een sprite te maken moet eerst een tekening van de sprite in een raster worden gemaakt. Figuur 2 is een voorbeeld hiervan. Deze tekening moet vervolgens worden omgerekend naar getallen die de computer begrijpt. Hierbij vormt elk byte in de tekening een specifiek getal. Dus van elk van de 63 bytes moet de waarde worden berekend. De 63 bytewaarden die hieruit komen zijn specifiek voor die sprite. In het kort gaat het berekenen van deze bytewaarden als volgt: de acht bits in een byte worden van rechts naar links genummerd. Het rechter bit heeft nummer 0 en het linker bit 7. Elk bit heeft een waarde, namelijk 2 tot de macht bitnummer (2 ^ nummer). Het meest linkse bit heeft bijvoorbeeld de waarde 2 ^ 7 = 128. De waarde van een byte wordt nu verkregen door de waarden van de bits die aan staan op te tellen. Hierbij staat een bit aan, als het bijbehorende pixel aanstaat. Zie voor een uitgebreidere uitleg voor het berekenen van bytewaarden de vorige aflevering van deze cursus. In figuur 2 staan naast de tekening de bijbehorende bytewaarden.

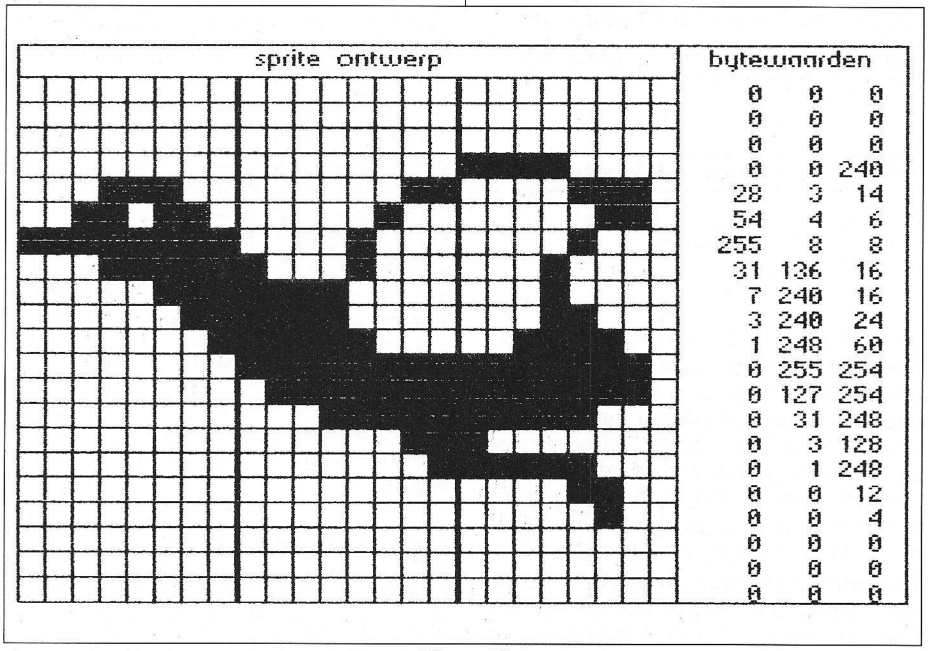
Opslaan van sprites

U heeft nu de sprite omgezet in getallen. Het volgende wat moet gebeuren is het doorgeven van deze getallen aan de computer. De bytewaarden moeten ergens in het geheugen worden gezet. De plaats waar u de bytewaarden in het geheugen zet, kunt u echter niet volkomen willekeurig kiezen. De eisen waaraan het geheugengebied moet voldoen, zijn de volgende:

* Het eerste adres van het geheugengebied moet een veelvoud van 64 zijn.

* Het geheugengebied moet tussen de adressen 0 en 16384 liggen (Dit heeft te maken met geheugenbanken, hierover echter meer in een latere aflevering).
* Het geheugen gebied tussen adres 4096 en 8192 kan niet gebruikt worden. Dit zal ook in een latere aflevering opgehelderd worden.

Door al deze eisen lijkt het mischien lastig om een geheugengebied te kiezen. In de praktijk blijkt dit echter erg mee te vallen. Als voorbeeld kiezen we het gebied dat begint op 832. (Dit is een veelvoud van 64, want 64 * 13 = 832). Aan de tweede en de derde eis wordt ook voldaan. De bytewaarden van de sprite moeten in de volgorde zoals in figuur 1 is aangegeven, in het geheugen worden gezet. Dus byte 0 komt op adres 832 en byte 62 op 894 te staan. Dit kan het best worden gedaan door alle bytewaarden in de goede volgorde in DATA-regels achteraan het programma te zetten. De bytewaarden kunnen dan door middel van een FOR-NEXT loop en de READ-instructie een voor een in het geheugen worden gezet.



Figuur 2

Spritepointers

Om nu de computer duidelijk te maken waar u uw sprites in het geheugen heeft geplaatst, moet u zogenaamde spritepointers gebruiken. Een pointer is het engelse woord voor wijzer. Een spritepointer wijst dus als het ware (door middel van een getal) naar het geheugengebied waar de sprite staat. Een spritepointer is een geheugenadres in de computer en de waarde van dit geheugenadres bepaalt waar de pointer naar toe wijst. De waarde van een spritepointer kan op de volgende manier berekend worden: het eerste adres van het geheugengebied waar de sprite staat, moet door 64 gedeeld worden. De uitkomst van deze deling is de waarde van de spritepointer. Dit verklaart waarom het geheugengebied moet beginnen op een adres dat een veelvoud is van 64. Als dit niet het geval zou zijn, zou de deling een breuk opleveren en breuken kunnen niet in een geheugenadres worden gezet. In ons voorbeeld is de waarde die de spritepointer moet bevatten dus 832 / 64 = 13.

Er is al gezegd dat er maximaal acht sprites tegelijk op het beeld kunnen worden gezet. Elk van deze sprites heeft een eigen spritepointer. De sprites worden genummerd van 0 tot en met 7 (Deze nummers zijn straks van belang). De spritepointers van deze acht sprites staan op de geheugenadressen 2040 (sprite 0) tot en met 2047 (sprite 7).

Sprites op het beeld zetten

Voor de sprite die u op het scherm wilt zetten moet u eerst een spritenummer kiezen. Dit moet een nummer zijn van 0 tot en met 7. Aan de hand van dit nummer kunt u de geheugenadressen bepalen die nodig zijn voor onder andere de positie op het scherm, kleuren en de spritepointer. Het is dus van belang dat u dit nummer goed onthoudt. Als voorbeeld kiezen we spritenummer 0 voor onze sprite uit figuur 2. De spritepointer is dus adres 2040. Hier poken we dus het getal 13 in. De computer weet nu dat sprite 0 op adres 832 begint.

Nu moeten we de positie op het scherm bepalen. Deze positie bestaat uit een x- en een y-coordinaat. Voor sprite 0 is het adres voor de x-coordinaat 53248 en voor de y-coordinaat 53249. Figuur 3 is een tabel met alle adressen die voor de sprites van toepassing zijn. Het scherm (het gedeelte binnen de borders) bestaat uit 320 x-coordinaten en 200 y-coordinaten. De linker bovenhoek heeft als coordinaten (24,50) en rechter onderhoek (343,249). Alle x-coordinaten die kleiner dan 24 of groter dan 343 zitten onder de zijborder en alle y-coordinaten kleiner dan 50 en groter dan 249 zitten onder de border. De sprite komt met zijn linkerbovenhoek precies op de gekozen coordinaten te staan. De rest van de sprite komt rechtsonder de gekozen coordinaten te staan. Dit houdt in dat met coordinaten (343,249) de sprite geheel onder de border staat behalve het pixel linksboven in de sprite. De gekozen y-coordinaat kan zonder meer in adres 53249 gepoked worden. De x-coordinaat geeft echter nog wat problemen, omdat een adres slechts waarden tussen 0 en 255 kan bevatten. We zullen ons voorlopig beperken tot de x-coordinaten die tussen 0 en 255 liggen. De grotere x-coordinaten zullen behandeld worden onder het kopje MSB. Met deze beperking kunnen we de x-coordinaat dus gewoon in adres 53248 poken.

Als laatste moet de sprite worden aangezet. Hiervoor dient adres 53269 (Dit adres wordt wel het sprite enable register genoemd). Elk van de acht bits in dit adres correspondeert met een spritenummer. Bit 0 correspondeert met spritenummer 0 etc. Het aanzetten van een sprite gebeurt door het correponderende bit aan te zetten. Voor sprite 0 gaat dit als volgt: POKE 53269,PEEK(53269) OR 2^0. In het algemeen kan een sprite worden aangezet met POKE 53269,PEEK(53269) OR 2^spritenummer. Het uitzetten gebeurt met POKE 53269,PEEK(53269) AND (255-2^spritenummer).

Het volgende programma zet de sprite uit figuur 2 twee keer op het scherm. Een keer in gewoon formaat en een keer vergroot. De pokes die worden gebruikt om de sprite te ver-

groten en te kleuren worden verderop uitgelegd. Maakt u zich dus geen zorgen als u het voorbeeld niet meteen helemaal snapt.

```
10 rem sprite voorbeeld
20 rem
30 poke 53280,0:poke 53281,
   0
40 rem zet bytewaarden in
   geheugen
50 for i=0 to 62:read a
60 poke 832+i,a:next
70 rem sprite-pointers
80 poke 2040,13:poke 2041,
   13
90 rem zet coordinaten
100 for i=0 to 3:read a
110 poke 53248+i,a:next
120 rem vergroot sprite 1
130 poke 53271,peek(53271) or
    2
140 poke 53277,peek(53277) or
    2
150 rem sprite kleuren
160 poke 53287,2:poke 53288,
    7
170 rem zet sprites aan
180 poke 53269,peek(53269)or
    3
200 rem bytewaarden
210 data 0,0,0,0,0,0
220 data 0,0,0,0,0,240
230 data 28,3,14,54,4,6
240 data 255,8,8,31,136,16
250 data 7,240,16,3,240,24
260 data 1,248,60,0,255,254
270 data 0,127,254,0,31,248
280 data 0,3,128,0,1,248
290 data 0,0,12,0,0,4
300 data 0,0,0,0,0,0,0,0,0
310 rem coordinaten
320 data 130,130,180,150
```

Kleuren

Elke sprite op het scherm kunt u afzonderlijk een van de zestien beschikbare kleuren geven. Voor de acht sprites zijn er acht kleuradressen, voor iedere sprite een. Deze adressen bevinden zich op de geheugenlocaties 53287 tot en met 53294. Adres 53287 is het kleuradres bij spritenummer 0 en 53294 bij spritenummer 7. In deze adressen kunt u de waarde van een kleur poken. POKE 53287,0 maakt bijvoorbeeld sprite 0 zwart.

Sprite adressen

Adres	Functie
53248	X-coordinaat sprite 0
53249	Y-coordinaat sprite 0
53250	X-coordinaat sprite 1
53251	Y-coordinaat sprite 1
53252	X-coordinaat sprite 2
53253	Y-coordinaat sprite 2
53254	X-coordinaat sprite 3
53255	Y-coordinaat sprite 3
53256	X-coordinaat sprite 4
53257	Y-coordinaat sprite 4
53258	X-coordinaat sprite 5
53259	Y-coordinaat sprite 5
53660	X-coordinaat sprite 6
53261	Y-coordinaat sprite 6
53262	X-coordinaat sprite 7
53263	Y-coordinaat sprite 7
53264	MSB register
53269	Sprite enable register
53271	Vertikale vergroting
53277	Horizontale vergroting
53287-53294	Kleur registers sprite 0 tot en met sprite 7
2040-2047	Sprite pointers sprite 0 tot en met sprite 7

Figuur 3

Bij het uitlezen van de kleuradressen doet zich een klein probleem voor. De hoogste vier bits (bit 4-7) staan namelijk altijd aan. De uitgelezen waarde is hierdoor altijd 240 groter dan de waarde van de kleur. Dit kan verholpen worden door de waarde te vergelijken door middel van AND met 15. Dus PEEK(53287) AND 15 (copyright de Boer) geeft de kleur van sprite 0.

MSB

Onder het kopje sprites op het scherm zetten hebben we ons beperkt tot x-coordinaten tussen 0 en 255. Nu zullen we uitleggen hoe sprites op x-coordinaten die groter zijn dan 255, kunnen worden neergezet. Dit kan worden gedaan met behulp van een extra bit, genaamd Most Significant Bit (MSB). Elke sprite heeft een eigen MSB. Deze acht MSB's zitten samen in adres 53264. Door het MSB van een bepaalde sprite aan te zetten, wordt aangegeven dat de x-coordinaat van die sprite hoger is dan 255. Als dit bit eenmaal is aangezet, kan de waarde van de x-coordinaat gewoon in het normale x-coordinaat adres worden gezet. Bij onze voorbeeld sprite was dit adres 53248. Er kunnen natuurlijk nog steeds geen waarden groter dan 255 in dit adres worden gezet, maar waarde 0 betekent nu (als het MSB bit aanstaat) dat de sprite op x-locatie 256 wordt gezet. Op deze manier kunt u x-coordinaten groter dan 255 aanspreken. De 8 MSB's van de acht standaard sprites zitten in volgorde van spritenummer in adres 53264 opgeslagen. Dit betekent dat als van sprite 0 het MSB moet worden aangezet, dit als volgt gedaan kan worden: POKE 53264,PEEK(53264) OR 1. Door het MSB weer uit te zetten, komt de spite weer op de normale x-coordinaten te staan. In de praktijk zal waarschijnlijk blijken, dat u de MSB's niet zo vaak gebruikt. Toch is het in sommige gevallen wel handig, bijvoorbeeld als u een pijltje over het hele scherm wilt laten bewegen, om iets aan te wijzen

Sprites vergroten

In feite is een sprite met zijn 504 pixels maar een klein figuurtje op het beeld. U kunt de sprites echter vergroten, met vergrotingsfactor 2. Een sprite kan zowel horizontaal als vertikaal vergroot worden. Dit kan afzonderlijk van elkaar gebeuren. Bij een horizontale en een vertikale vergroting, wordt de sprite dus vier keer zo groot. Het nadeel echter hiervan is, dat de sprite veel groffer wordt. Want

elk pixel wordt nu twee keer zo breed en/of twee keer zo lang. Voor beide vergrotingen bestaat er een apart adres in de VIC. Voor de vertikale vergroting is adres 53271 gereserveerd en voor de horizontale 53277. De acht bits in deze adressen zijn hetzelfde geordend als de bits in het sprite enable register en MSB register. Door bijvoorbeeld bit 0 van adres 53271 aan te zetten, wordt sprite 0 twee keer zo groot in vertikale richting.

Animatie van sprites

Animatie met sprites is redelijk simpel te bewerkstelligen. Dit komt natuurlijk omdat sprites aparte adressen hebben voor de locatie op het beeldscherm. Het is daardoor niet moeilijk om een sprite over het scherm te laten bewegen. Het enige wat hiervoor gedaan moet worden, is het veranderen van de x- en y-coordinaat van de desbetreffende sprite. Dit kan bijvoorbeeld in een FOR-NEXT loop worden gedaan, waarbij bij iedere iteratie de waarden van de coordinaat-adressen worden verhoogd of verlaagd. U moet er echter wel op letten dat, zodra de x-coordinaat de waarde 255 overschrijdt het MSB moet worden gezet. Ook kunnen de waarden van deze coordinaat-adressen worden gekoppeld aan de joystick. De joystick kan (in port 2) worden uitgelezen in adres 56320. Aan de hand van de richting van de joystick (oftewel de waarde van adres 56320) kunnen x- en y-coordinaat worden aangepast. Bij het bewegen van een sprite kan ook handig gebruik worden gemaakt van de coordinaten die zich achter de border bevinden. Door bijvoorbeeld een sprite geheel achter de bovenborder te plaatsen en deze vervolgens langzaam naar beneden te bewegen, verschijnt de sprite stapje voor stapje op het scherm en niet in een keer.
Met de hierboven gegeven voorbeelden kunt u een sprite normaal over het beeld laten bewegen. Het wordt echter pas leuk als de sprite tijdens dit bewegen zelf ook verandert. Een voorbeeld hiervan is een poppetje dat tijdens het voortbewegen over het scherm ook echt zijn benen meebeweegt. Om dit voor elkaar te krijgen, moet u van te voren eerst twee of meerdere sprites ontwerpen. In elk van deze sprites moet de stand van het poppetje iets veranderd zijn. Als dit gedaan is, moet u deze sprites allemaal ergens in het geheugen zetten (natuurlijk niet allemaal op dezelfde plek). Daarna kiezen we een sprite uit, zeg sprite 0. Van deze sprite moeten eerst de coordinaten goed worden gezet (bijvoorbeeld (100,100)) en daarna moet de sprite nog worden aangezet. Door nu de spritepointer van sprite 0 (adres 2040) steeds naar een andere ontworpen sprite in het geheugen te laten wijzen, verandert de sprite op het beeld. Door dit snel genoeg te doen en als de sprites slechts een klein beetje van elkaar verschillen ontstaat een vloeiende beweging.
U heeft dus een paar sprites in het geheugen staan en moet daarvan de waarden berekenen die in de spritepointer moeten worden gezet. Deze waarden werden verkregen door het eerste geheugenadres waar de sprite staat door 64 te delen. Deze waarden moet u om de beurt in de spritepointer van sprite 0 poken. Ook dit kan in een FOR-NEXT loop gebeuren. Door daarbij ook de locatie van de sprite te veranderen, krijgt u een echt bewegend figuurtje.

Applicatie programma's

Als laatste geven we nog twee applicatie programma's. De eerste is een eenvoudige sprite-editor. Hiermee kunt op een gemakkelijke manier sprites ontwerpen en bytewaarden laten uitrekenen. Om een sprite te ontwerpen, moet u met de joystick in poort 2 een cursor over het scherm sturen. Met deze cursor kunt u tekenen, wissen en gewoon naar een andere positie op het scherm gaan. Als de cursor een kruisje is, kunt u wissen. Als de cursor een bolletje is, kunt u tekenen. En als de cursor een plusje is, kunt u de cursor gewoon bewegen zonder dat er iets aan het ontwerp verandert. U kunt door een aantal malen op de vuurknop van de joystick te drukken de juiste cursor uitkiezen. De rest van de bediening wijst voor zich.

Animatie

Het tweede programma vereenvoudigt het animeren van sprites. In Basic is het lastig om met constante snelheid de vorm van de sprites snel te veranderen, waardoor de animatie vaak schokkerig wordt. Met dit machinetaal programma kunt u sprites echter probleemloos vloeiend bewegen.
Als eerste moet u het programma runnen. Aan de hand van een voorbeeld zullen we laten zien hoe u een figuurtje in sprite 0 kunt laten bewegen. Als eerste tekent u een aantal verschillende standen van het figuurtje en slaat deze allemaal in het geheugen op. Daarna zet u sprite 0 op de juiste plaats op het scherm. Vervolgens moet aan de computer worden doorgegeven welke standen achtereenvolgens op het scherm moeten worden gezet, zodat animatie ontstaat. Hiervoor is voor elk van de acht sprites een tabel van 32 bytes beschikbaar. De tabellen van sprites 0 tot 7 beginnen op de adressen 52992, 53024, 53056, 53088, 53120, 53152, 53184, 53216. In ons voorbeeld hebben we dus de tabel nodig, die op 52992 begint. Eerst bepalen we van alle opeenvolgende standen van het figuurtje de spritepointers. Deze pointers moeten dan achter elkaar in de tabel worden gepoked. Als er bijvoorbeeld 3 verschillende standen zijn met spritepointers 13, 14 en 15, dan moeten deze in de adressen 52992 tot 52994 worden gepoked. Achter deze pointers moet een 0 in de tabel worden gepoked. In ons voorbeeld dus in adres 52995. De computer weet dan wanneer alle figuurtjes aan de beurt zijn geweest. Tenslotte moet nog de snelheid van de animatie aan de computer worden gegeven. Hiervoor zijn de adressen 52984 tot 52991 voor sprites 0 tot 7 nodig. De snelste snelheid is 1 en de langzaamste is 255. Een snelheid 0 betekent dat de sprite niet beweegt. Als we dus het figuurtje in sprite 0 met snelheid 4 willen laten bewegen, moeten we in adres 52984 de waarde 4 poken. De computer zal nu een voor een de verschillende standen van het figuurtje op het scherm zetten. Na de laatste stand wordt weer de eerste stand op het scherm gezet, zodat sprite 0 continu blijft bewegen. Om de beweging te stoppen moeten we een 0 in adres 52984 poken. Met dit programma kunt u zo acht verschillende figuurtje tegelijk animeren.
Na STOP/RESTORE of reset is de machinetaal-routine uitgeschakeld. Om deze routine weer te activeren, moet SYS 49152 worden gebruikt. Veel succes ermee.

Als hulp bij het programmeren van sprites, hebben we in figuur 3 alle behandelde sprite adressen met een korte functie aanduiding bij elkaar gezet. In de volgende aflevering zullen we meer over sprites vertellen. We zullen het dan onder andere hebben over overlappen van sprites, multi-color sprites en botsingen tussen sprites. En als truc de la truc sprites in meer dan 100! kleuren.

Michel de Boer en Hylke Spranger

Listing 1.

```
10 rem ******************
20 rem *                *
30 rem * sprite - editor *
40 rem *                *
50 rem ******************
60 rem
70 rem initialisatie
80 rem
90 poke 53281,254:poke 53280,254
100 u1(1)=86:u1(2)=81:u1(3)=43:u2(1)=32
110 u2(2)=160:u2(3)=32:dim js(16)
120 for x=6 to 15:read a:js(x)=a:next
130 for x=0 to 7:poke53287+x,0
140 poke 2040+x,248+x:next:for x=0 to 97
150 read a:poke49152+x,a:next
160 rem
170 rem sprite kiezen
180 rem
190 printchr$(147)chr$(5);
200 xc=1079:qq=32:cu=1
210 print"welke sprite veranderen (0-7)"
220 input sn:if sn<0 or sn>7 then 210
230 ga=(248+sn)*64:h=int(ga/256):a=sn*2
240 l=ga-256*h:poke 253,l:poke 254,h
250 poke 53248+a,50:poke 53249+a,150
260 print tab(40)"joystick in port 2"
270 rem
280 rem schermopbouw
290 rem
300 for q=0 to 2000:next:print chr$(147)
310 for q=1038 to 1063:pokeq,90
320 poke q+880,90:next
330 for q=1038 to 1918 step 40
340 poke q,90:poke q+25,90:next
350 print"1: sprite af"
360 print"2: beeld leeg"
370 print"3: beeld vol"
380 print"4: x,y klein"
390 print"5: x groot"
400 print"6: y groot":poke 53269,2^sn
410 rem
420 rem joystick & toetsenbord uitlezen
430 rem
440 get a$:sys49152:poke xc,u1(cu)
450 for t=0 to 70:next:if a$="1" then 610
460 if a$="2" then op=32:gosub 680
470 if a$="3" then op=160:gosub 680
480 if a$="4" then poke 53271,0:poke 53277,0
490 if a$="5" then poke 53277,2^sn
500 if a$="6" then poke 53271,2^sn
510 a=peek(56320):ri=js(abs(a-111))
520 if a=111 then 560
530 if ri=0 or peek(xc+ri)=90 then 440
540 u2(3)=qq:qq=peek(xc+ri):xc=xc+ri
550 poke xc-ri,u2(cu):goto 440
560 cu=cu+1:if cu=4 then cu=1
570 goto 440
580 rem
590 rem opties uitvoeren
600 rem
610 poke xc,qq:sys49152
615 print chr$(147):poke 53269,0
620 for x=0 to 20:printx":";
630 for y=0 to 2:print peek(x*3+y+ga);
640 next:print:next
650 print tab(40)"druk een toets in."
660 get a$:if a$="" then 660
670 goto 190
680 a=1079:for x=0 to 20:for t=0 to 23
690 poke a+t,op:next:a=a+40:next
695 qq=op:return
700 rem
710 rem data getallen
720 rem
730 data 41,-39,1,0,39,-41,-1,0,40,-40
740 data 169,0,141,252,207,169,4,133
750 data 252,169,55,133,251,160,0,169
760 data 0,141,253,207,162,7,177,251
770 data 201,160,208,21,138,72,169,1
780 data 224,0,240,5,10,202,76,32
790 data 192,13,253,207,141,253,207,104
800 data 170,202,200,224,255,208,223,152
810 data 72,172,252,207,173,253,207,145
820 data 253,104,168,238,252,207,192,24
830 data 208,197,24,165,251,105,40,133
840 data 251,165,252,105,0,133,252,201
850 data 7,208,178,165,251,201,127,208
860 data 172,96
```

Listing 2.

```
10 rem ******************
20 rem * sprite animator *
30 rem ******************
40 rem
50 for i=0 to 118:read a
60 s=s+a:poke 49152+i,a
70 next
80 if s<>16888 then print"fout in data"
90 sys 49152
100 data 162,7,169,0,157,127,192,157
110 data 248,206,169,1,157,119,192,202
120 data 16,240,120,169,54,141,20,3
130 data 169,192,141,21,3,169,127,141
140 data 13,220,169,1,141,26,208,169
150 data 250,141,18,208,173,17,208,41
160 data 127,141,17,208,88,96,169,1
170 data 141,25,208,169,0,133,251,169
180 data 207,133,252,162,0,189,248,206
190 data 240,30,222,119,192,208,25,189
200 data 248,206,157,119,192,188,127,192
210 data 177,251,208,6,157,127,192,168
220 data 177,251,157,248,7,254,127,192
230 data 24,165,251,105,32,133,251,232
240 data 224,8,208,209,76,49,234
```



Het is alweer even geleden, maar hier zijn we weer. Een continuering van onze cursus Machinetaal. Tjipke van der Land gaat op zijn geheel eigen wijze door met het werken aan de 65XX processor die onze Commodore zo'n heel eigen karakter geeft.

C-64 Machinetaal

Deel 13: indirecte adressen

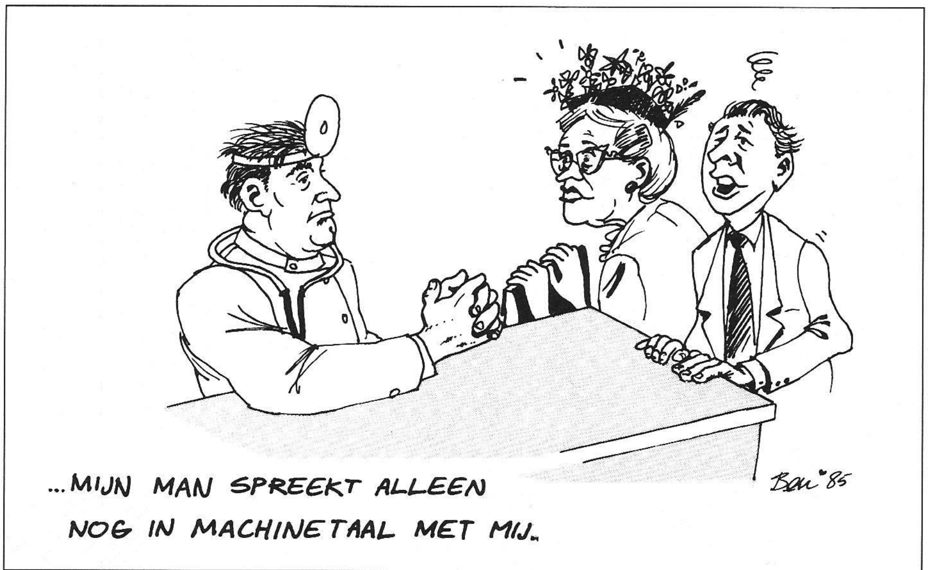
Omdat we even afwezig zijn geweest gaan we deze keer er weer goed tegenaan. We zullen deze keer weer flink veel instructies doornemen. Maar voor we dat gaan doen, zullen we even terugblikken naar de vorige lessen in deze cursus. En dan kunnen we concluderen dat we geen onbekenden meer zijn op gebied van machinetaal. Als alles goed is zijn we op dit moment goed in staat te begrijpen hoe een microprocessor schematisch werkt. En uiteindelijk hebben we toch al heel wat programmaatjes gemaakt, die op zich aardig werken.

Indirecte adressen

De vorige keer zijn we gebleven bij de moeilijke geïndexeerd indirect X en de Indirect geïndexeerd Y Instructies, wat je misschien nog steeds niet goed begrijpt. Al met al heb je in de tussenliggende periode voldoende tijd gehad om flink te kunnen oefenen met deze instructies, en krijg je nog ruimschoots de kans om nog meer te oefenen, alleen al in de komende lessen, want een groot gedeelte van de machinetaal instructies hangt samen met deze vormen van adressering. Het enige wat we tot nu toe gedaan hebben is het laden van de accu langs de geïndexeerd indirect X en de indirect geïndexeerd Y manier. Het is dus niet zo dat het laden van de accu het voorrecht heeft om als enige gebruik te mogen maken van deze moeilijke maar zeer bruikbare instructie-methode. Er zijn veel meer instructies die langs deze weg zichzelf data en adressen kunnen verschaffen.

Subroutine

Voordat we daar mee door gaan, gaan we eerst terug naar het idee om het printen van strings te verwerken in een subroutine die telkens teksten ophaalt uit een datablok, waar deze teksten in staan opgeslagen, en vervolgens afdrukt op de plek in het videogeheugen (het scherm dus) waar je de tekst zichtbaar wilt hebben. Daarom laten we deze moeilijke instructies voor het moment even links liggen om het begrip subroutine wat nadere onder de loep te nemen. Wat is eigenlijk een subroutine? Voordat we verder kunnen met programmeren zullen we ons weer wat theoretische begrippen moeten eigene voordat we hiermee kunnen gaan werken. Een subroutine is wat hij met zijn benaming al een beetje duidelijk maakt, hij voerde een (SUB) programma uit. Dus in begrijpelijk Nederlands, hij springt uit een lopend programma naar een losstaand programma, wat op zijn beurt iets uitvoert ten behoeve van het hoofdprogramma, waar hij uitgesprongen is, bijvoorbeeld de stand van een berekening op het scherm afdrukken. Stel dat een programma bezig is met een heel gecompliceerde berekening of uitvoering van een proces. Inmiddels zal je uit ervaring met de computer wel eens iets meegemaakt hebben in de trant van "Doet ie nog wat of zit hij vast" en al na een stevig halfuurtje begin je wat te twijfelen, en zet je hem toch maar uit. Om dit te voorkomen en je wat meer zekerheid te verschaffen over het nog bezig zijn van de processor, kan je tijdens de berekening of het verloop van het proces af en toe iets afdrukken op het scherm waardoor je weet dat hij inderdaad nog druk bezig is, met het uitvoeren van allerlei belangrijke opdrachten. Dit af en toe iets afdrukken kan je laten doen door een subroutine of subroutine's. Hoe gaat dat in zijn werk? Als een programma bezig is met iets uit te voeren komt het programma niet automatisch een keer per seconde op het idee om maar eens een subroutine uit te gaan voeren. Het is wel jammer, maar de techniek is op dit moment nog niet zover dat een computer uit zichzelf iets gaat doen.

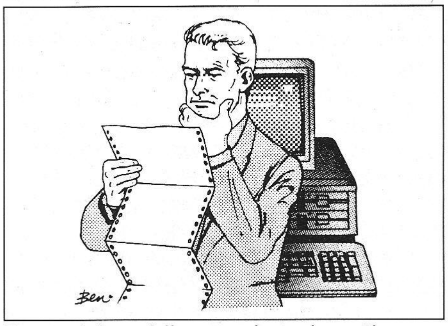


JMP instructie

Het vervelende relaas is dan ook dat je dat zelf moet aangeven in een programma, wanneer je een subroutine uit wilt laten voeren. De vraag naar het "Hoe" zullen we nu wat beter bekijken. De instructie om te springen naar een subroutine lijkt sprekend op een instructie waar we al wat mee te maken hebben gehad, namelijk de JMP instructie.
Dit is immers een instructie waar je een sprong mee kan maken binnen een programma. Een voorwaarde voor deze sprong is wel dat je hem op de goede plek laat springen, want als je hem laat springen naar een plek waar geen instructies staan, hoef je niet een melding te verwachten van "ho ho verkeerde sprong", want dan loopt hij gewoon vast.
De instructie voor het springen naar een subroutine is JSR. De afkorting JSR (Jump to Sub Routine) geeft al min of meer weer wat staat te gebeuren. Een belangrijke vraag is echter wat het effectieve nut is van een subroutine, en wanneer je een subroutine precies laat uitvoeren. Een subroutine heeft als nut je programma meestal (aanzienlijk) korter en op z'n minst overzichtelijker te maken.
Bijvoorbeeld als je in een programma in totaal 300 teksten moet printen. Als je niet de beschikking hebt over een JSR instructie, zul je 300 keer een programmadeel moeten schrijven om de bedoelde tekst op zijn plaats te krijgen. Je zult begrijpen dat je bij het 40ste identieke programmadeel het al redelijk zat bent, zo erg zelfs dat je de overige 260 teksten er maar bij gaat denken en dat is niet de bedoeling. Daarom is de instructie JSR ook een welkome instructie op de lijst van instructies voor luie wezens.

Actie

Belangrijk is het te weten wat precies gebeurt bij het uitvoeren van deze instructie. Om te beginnen spring je met JSR uit het hoofdprogramma naar de bedoelde subroutine, bijvoorbeeld : JSR $2000. Eenmaal aangekomen bij het eerste adres van deze routine gaat deze de opdrachten uitvoeren zoals in een normaal programma, dus het laden van getallen in de accu en noem maar op. Aan het eind van deze subroutine gebruiken we een voor ons nieuwe instructie namelijk de RTS (ReTurn from Subroutine). Deze instructie zorgt ervoor dat je weer terug komt in het hoofdprogramma vlak na de instructie JSR. Een leuk ding om te weten is hoe de processor precies kan onthouden waar hij terug moet komen na het uitvoeren van een subroutine. Dit begint eigenlijk al op het moment waar je hem laat springen naar de subroutine. Zodra de processor de JSR opdracht heeft geïnterpreteerd, gaat hij namelijk het huidige instructieadres + 2 opslaan op de STACK.



Dan springt hij naar de subroutine en gaat de opdrachten uitvoeren die verlangd worden. Aan het eind van deze routine, dus wanneer hij weer terug moet naar het hoofdprogramma vinden we aan het eind de instructie RTS. Deze instructie doet in wezen niets anders dan het oude adres van de stack halen. Zodra de waarde van de stack weer in de programma teller geladen is, verhoogt deze zichzelf automatisch met 1 en vervolgt het verdere instructie verloop.
Om nog even terug te komen op wijze waarop de processor de waarde van de programma-teller op de stack zet: dit gebeurde door de programmateller-waarde plus 2 op de stack te zetten. Waarom gebeurt dat eigenlijk? Dit is zo, omdat de instructie JSR alleen maar absoluut kan adresseren. Dus met een volledig adres van twee bytes. Dit heeft automatisch tot gevolg dat deze hele instructie uit 3 byte's bestaat. Dus vanaf het oude adres zou de programmateller precies 3 bytes verder moeten om het programma te kunnen vervolgen. Als we dan even gaan rekenen zul je zien dat dit exact klopt. Bij de instructie RTS haalt hij de waarde van de programmateller van de stack. Deze waarde was bij het plaatsen op de stack eerst verhoogd met 2, dus in wezen al met 2 bytes extra opgeslagen. Zodra deze waarde is terug gelezen in de programmateller (pc) verhoogt deze zichzelf met 1, en dat was nog de ontbrekende byte, om probleemloos door te kunnen gaan.
Door middel van een simpel voorbeeldje zullen we laten zien hoe het ongeveer in zijn werk gaan om een subroutine aan te roepen. (zie fig. 1)

Als je hier goed naar kijkt, zal het je opvallen dat de accu of registers niet worden beïnvloed door het springen naar een subroutine. Dus waar je hier rekening mee moet houden, is dat bij het onderbreken van een lopend proces naar een gecompliceerde subroutine waar veel gebruik word gemaakt van de accu en andere registers, bij terugkeer in het hoofdprogramma deze waarden wel eens helemaal verkeerd terug kunnen komen. In deze gevallen moet je eerst de registers die je wilt gaan gebruiken op de stack zetten, in dit voorbeeld de instructie's PHA en PLA. Wat hier precies mee bedoeld wordt komen we op terug.

:Hoofdroutine

	MNEMONIC	**COMMENTAAR**
.a 1000	LDA #$32	Accu laden met eerste getal
	LDX #$00	X register op nul zetten
	STA $0578,X	1 Waarde accu in videogeheugen
	PHA	Waarde accu op stack zetten
	JSR $1500	Naar subroutine springen
	PLA	Waarde accu van stack halen
	INX	X register voor 2 keer verhogen
	STA $0578,X	3 Waarde accu in videogeheugen
	BRK	

:Subroutine

	MNEMONIC	COMMENTAAR
.a 1500	LDA #$36	Accu laden met tweede getal
	INX	X register voor 1 keer verhogen
	STA $0578,X	2 Waarde accu in videogeheugen
	RTS	Terugspringen na JSR opdracht

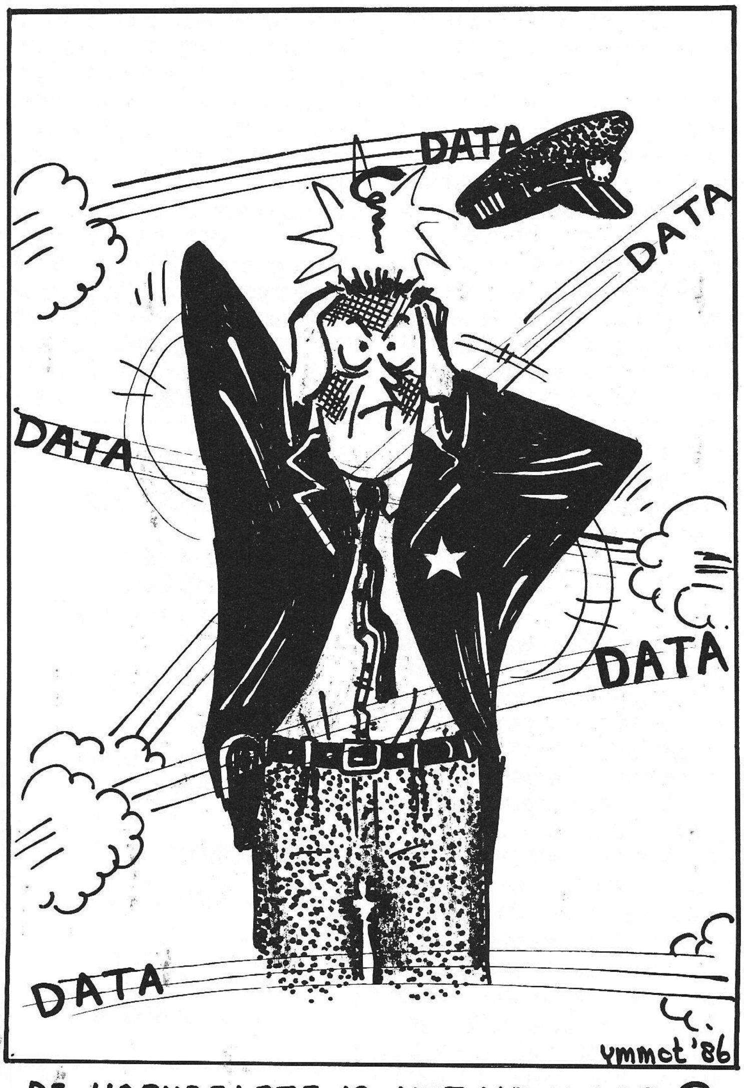
fig. 1

Registers

Wat belangrijk is om te weten, is dat je niet zomaar een subroutine kunt aanroepen zonder te denken aan het *lopende hoofdprogramma wat onder*broken wordt. Tenslotte gebruikt het hoofdprogramma de registers, die ook de subroutines gaat gebruiken. Een klein vergelijkbaar onderdeeltje is het op nul zetten van X- en Y-register, voordat we de registers gaan gebruiken, want er kan immers nog een foutieve waarde in zitten van een voorgaand programma, of andere subroutine. Daarom zullen we ook een aantal voorbeelden aansnijden om dit verschil duidelijk te maken.
Voordat we dit goed uit kunnen leggen door middel van voorbeelden is kennis vereist van een aantal extra instructies. Deze instructies die we nodig hebben om het verschil tussen diverse subroutines te begrijpen hebben betrekking op het tijdelijk bewaren van gegevens, die we in het latere verloop van het programma weer nodig hebben. En van de vragen is dan hoe je tijdelijk gegevens op kunt slaan, en hoe ze weer terug te vinden zijn.
Met dit probleem zullen we beginnen. De registers waar meestal belangrijke gegevens in staan zijn de accu, X- en Y-register, niet te vergeten de statusvlaggen die aangeven hoe de situatie is van een berekening is.

Voorbeeld

Bijvoorbeeld stel dat het resultaat van een berekening in de accu komt, en het antwoord past niet in de accu. Dit probleem is ooit eens in 1 van de eerste lessen behandeld (zo zie je maar weer dat je alles nodig blijft houden), door nadere studie van deze uitgaven komen we erachter dat dan de processor het overflow bit gaat zetten. Dus als je na het plaatsen van het antwoord in de accu gaat springen naar een subroutine, die ook fiks aan het werk gaat met de accu, en je slaat voor het springen naar de subroutine alleen het antwoord van de accu op, kom je bedrogen uit als deze waarde weer terug wordt gehaald voor een verder vervolg van de berekening.
Dus is het belangrijk om in dit geval ook de statusvlaggen op te slaan. Gelukkig voor ons heeft de 65xx daar standaard een aantal instructies voor die in een 1 byte instructie een aantal belangrijke gegevens kan wegschrijven. We gaan uitsluitend de instructies behandelen die door de standaard 65xx worden afgewerkt, want er zijn namelijk speciale uitvoeringen van de 65xx aangeduid met 65Cxx die over meer instructies op dit gebied beschikken dan de standaard 65xx. Deze extra instructie gaan we niet behandelen, want bijna niemand van onze Commodore bezitters beschikt over zo'n processor. De instructies die wel door onze 65xx worden begrepen zijn de PHA, PHP, PLA en PLP, waarvan we er net al een paar genoemd hebben. De instructie PHA (PusH A) stopt de waarde van de Accu op de stack, en de instructie PLA (PuLl A) haalt de waarde van de stack en plaatst hem weer in de accu. Voor de instructie PHP (PusH Processor status) geld hetzelfde als bij PHA alleen word hier de situatie van de vlaggen op de stack gezet. En tenslotte voor de PLP (PuLl Processor status) die de toestand van de vlaggen weer van de stack haalt. In het hierboven staande voorbeeld staan de zojuist behandelde instructie's al verwerkt. Bij dit voorbeeld zie je dat we een waarde laden in de accu, deze vervolgens zichtbaar te maken door hem op te slaan in het video geheugen. Dan wordt de waarde van de accu op de stack gezet, en de accu vervolgens geladen met een ander getal, wat ook in het videogeheugen word opgeslagen.
Dit doen we, zodat je kunt zien dat er inderdaad een ander getal in de accu heeft gestaan. Vervolgens halen we de oorspronkelijke waarde van de accu terug van de stack, en drukken ook dit af in het video geheugen. Als het goed is moet dus de eerste waarde die afgedrukt is in het videogeheugen gelijk zijn aan de derde waarde. Zo krijg je ongeveer een indruk hoe dit in z'n werk gaat. Dit is belangrijk om te weten, omdat je soms ook waarde's wilt doorgeven aan de subroutine, waarmee je de subroutine wilt aangeven waar hij bijvoorbeeld indirect andere waarden kan halen. Denk bijvoorbeeld maar eens aan ons voorbeeld in de vorige uitgaven, waar we Indirect geïndexeerd Y een string in



..DE HOEVEELSTE IS HET VANDAAG ?...

het videogeheugen hebben geplaatst. Het startadres van deze string kan je door middel van de accu doorgeven aan de subroutine. Belangrijk is dan dat de oorspronkelijke waarde van accu op de stack wordt gezet, waardoor je hem na de subroutine gewoon terug kunt halen.

Programmeer-opdracht

Iets wat ook belangrijk is binnen een subroutine, om zoveel mogelijk akties met het X- of Y-register en de accu af te ronden om niet in de verwarring te raken bij het verlaten van de subroutine als je weer met deze zelfde registers door moet werken. Dus het doorgeven van waarden en het terughalen van waarden in een subroutine moet je eigenlijk zoveel mogelijk het gebruik van registers vermijden. Je kunt dat wel gaan doen als je een speciaal stukje geheugen voor pakt om eventueel uitwisselbare waarden in te plaatsen. Je hebt net kunnen zien dat het opslaan van eenvoudige zaken zoals de accu en statusvlaggen met een standaard instructie kan worden verzorgd. Voor het geval je andere gegevens wilt opslaan, zoals het Y-register en het X-register, dan moet je je toevlucht nemen tot je eigen programmeerkunst om dit veilig te bewaren.

Een leuke opdracht voor de volgende keer is dan ook dat je zelf een programma maakt waarin je een subroutine aanroept. Voordat je de subroutine aanroept moet je echter alle registers opslaan, dus Accu, X- en Y register, de statusvlaggen en noem maar op. Als je terugkomt uit de routine moet je ze uiteraard weer terughalen. Om je een klein beetje op weg te helpen krijg je een paar tips.

Voor het opslaan van de accu en de statusvlaggen heb je een paar instructie beschikbaar. En voor het opslaan van het X en het Y register moet je maar gaan experimenteren met de instructies STX en STY. Deze instructies werken precies als de instructie STA, alleen de laatste letter staat voor het register wat je wilt opslaan. Elke keer na het opslaan van de registers, moet je ze maar eens laden met een andere waarde, wat je in wezen in de subroutine kunt doen.

De bedoeling hiervan is dat je kunt zien of je de juiste waarde wel terug haalt. Het terughalen van het X- en het Y- register doe je met de instructie's LDX en LDY die precies werken als de LDA instructie. De volgende keer beginnen we met het uitwerken van de opdracht die je nu zelf moet gaan maken. Bedenk wel, hoe meer je voor jezelf uitprobeert des te beter begrijp je hoe het allemaal werkt.

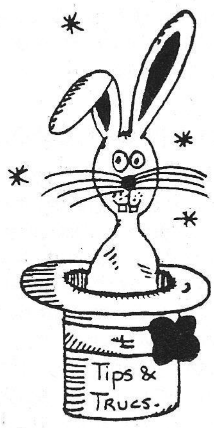
Voor deze keer laten we het bij deze informatie, omdat je voldoende stof hebt om te oefenen. Vergeet de stof van de vorige keren niet want die komt sneller terug dan je denkt, met de geïndexeerde en de indirect methoden.

Oefen veel met deze nieuwe instructies, met name de push en pull instructie's om te kijken wat je er allemaal mee kan doen. Voor de volgende keer gaan we een gecombineerd gebruik maken van deze instructie's, maar voordat het zover is moet je ook hiermee flink oefenen om niet bij het combineren van deze instructies het spoor bijster te raken. Dus doe je best, want het begint nu echt interessant te worden, tot de volgende keer maar weer, en de groeten van CHIP.

Tjipke van der Land

Deze keer behandelen Hylke Sprangers en Michel de Boer vier tips. De eerste tip verbetert de Basic RESTORE-instructie. De tweede en de derde tip zijn vooral bedoeld voor de machinetaal programmeur en de laatste tip gaat over een geliefd onderwerp: de klok.

TIPS & TRUCS 64



Restore met regelnummer

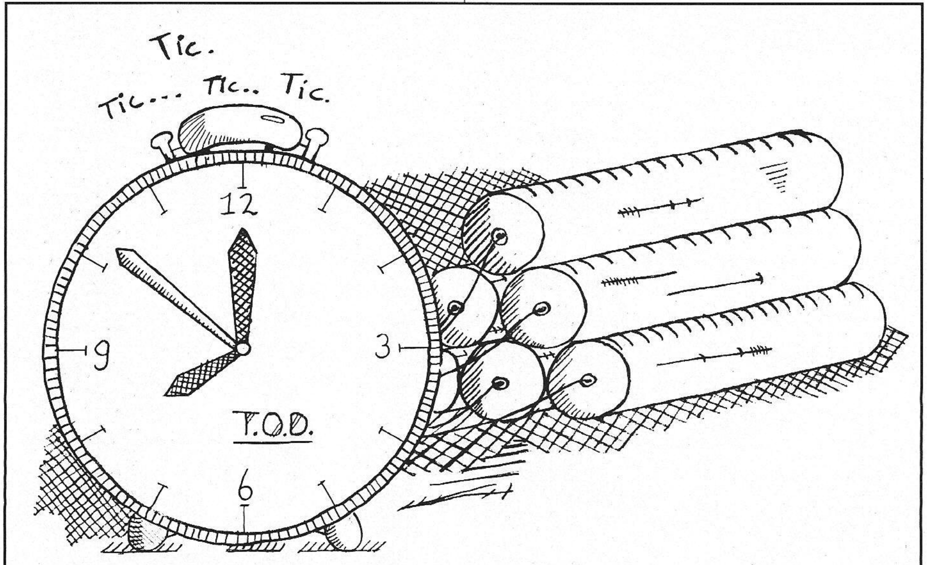
Met de DATA-instructie kunnen gegevens, in de vorm van getallen of tekststrings, in een programma worden gezet. Deze gegevens kunnen dan een voor een door middel van een READ-opdracht in een variabele gelezen worden. Als de gegevens ingelezen worden, wordt intern een data-pointer bijgehouden, waar de computer is met lezen van de gegevens. Dus steeds als er iets gelezen is, wordt deze pointer verhoogt, zodat hij naar het volgende gegeven wijst. Met de instructie RESTORE kan deze pointer weer teruggezet worden naar het eerste gegeven in de DATA-lijst. Zo kan de informatie in de DATA-statements weer opnieuw gelezen worden. De Commodore 64 heeft standaard echter niet de mogelijkheid om van een willekeurige data-regel een gegeven in te lezen. U kunt dus niet een regelnummer meegeven met de RESTORE-instructie, om dan vervolgens van die data-regel iets in te lezen (op sommige andere computers kan dit wel, bijvoorbeeld: RESTORE 100. Met een READ instructie wordt daarna het eerst volgende gegeven van regel 100 gelezen).
Er bestaat een manier om dit op te lossen. De computer moet dan verteld worden, waar het gegeven staat dat als eerste gelezen moet worden. Er is echter een beperking: het gegeven kan niet midden in een data-statement staan, maar vooraan de regel staan. De data-pointer moet dan onder andere veranderd worden. Met het volgende machinetaal-programma kunt u een RESTORE met regelnummer imiteren. Eerst moet de machinetaal-routine in het geheugen gezet worden op adres 49152. Als u dan bijvoorbeeld van regel 70 iets wil lezen, moet u de hi- en lo-byte van 70 in de zero-page adressen 64 en 63 zetten. Dit gaat als volgt: hi-byte = int(70/256) en lo-byte = 70-hi*256. Daarna moet de machinetaal-routine aangeroepen worden (SYS 49152). Het tweede programma geeft hier een voorbeeld van.

```
10 rem restore met
   regelnummer
20 for x = 0 to 23:read i
30 poke 49152+x,i:next
40 data 165,63,133,20,165
50 data 64,133,21,32,19,166
60 data 165,95,233,1,133,65
70 data 165,96,233,0,133,66,
   96
```

```
10 rem voorbeeld
20 rn = 70:hi = int(rn/256)
30 lo = rn-hi*256:poke 63,lo
40 poke 64,hi:sys 49152
50 read a$:print a$
60 data artvark,noot,mies
70 data commodore
```

TOD klok

Er zijn veel toepassingen waarbij een tijd moet worden bijgehouden (bijvoorbeeld de denktijd in een schaakspel). In Basic kan dit worden gedaan via de variabelen TI$ en TI. Deze variabelen worden zestig maal per seconde bijgewerkt door de interrupt-routine. Helaas wordt deze interrupt-routine tijdens het laden van en saven op tape of disk vertraagd, waardoor de klok gaat achterlopen. De klok gaat ook achterlopen, als er zelfgeprogrammeerde interrupts worden gebruikt. In deze gevallen zijn de variabelen TI$ en TI niet geschikt voor het bijhouden van de tijd. Gelukkig beschikt de Commodore nog over een andere klok, die direct door een interne klok wordt bijgehouden en niet door de interrupt-routine, zodat de twee bovenbeschreven problemen zich niet meer voordoen. Deze klok wordt de Time Of Day (TOD) klok genoemd.
Deze klok houdt de tijd in uren, minuten, seconden en tiende seconden bij. Ook wordt bijgehouden of de tijd AM of PM is. Het is zelfs mogelijk om een alarmtijd te zetten.
De TOD klok bevindt zich op de geheugen adressen 56328 tot en met 56331. In figuur 1 staat wat er in deze adressen wordt bijgehouden. De waarden van deze adressen kunnen echter niet zonder meer worden uitgelezen. Deze waarden staan namelijk



in het zogenaamde Binary Coded Decimal (BCD) formaat. (Zie voor uitgebreide uitleg de tip over BCD getallen.) Feitelijk komt het erop neer dat PEEK(adres) AND 15 het rechter cijfer van de decimale waarde oplevert en (PEEK(adres) AND 240)/16 het linker cijfer van de decimale waarde oplevert. Bij de uren is er echter een kleine uitzondering. Zoals in figuur 1 te zien is, wordt alleen bit 4 in plaats van bits 4-7 gebruikt voor het linker cijfer. Dus hier moet (PEEK(56331) AND 16)/16 worden gebruikt om het linker cijfer te bepalen. Door bit 7 van adres 56331 te testen kan bepaald worden of de tijd AM dan wel PM is. Als de uitkomst van PEEK(56331) AND 2^7 gelijk is aan 128 dan staat bit 7 aan en is de tijd dus PM anders is de tijd AM. Bij het schrijven van deze adressen moet niet de gewone waarde worden gepoked maar 16*linkercijfer + rechtercijfer.
Er is een probleem bij het uitlezen van de tijdadressen. Stel bijvoorbeeld dat de tijd 08:59:59 is. Na het uitlezen van de uren kan de klok overspringen op 09:00:00 zodat bij het uitlezen van de minuten de waarde 0 wordt verkregen, waardoor u zou denken dat de tijd 08:00:00 is. Dit probleem is door de makers van Commodore opgelost door te stoppen met het veranderen van de tijdadressen, nadat het adres voor de uren is uitgelezen. Intussen loopt de interne klok wel door. Pas nadat de tiende seconden zijn uitgelezen, worden de tijdadressen weer bijgehouden. Als u de tijdadressen wilt lezen moet u dus als eerste de uren lezen en als laatste de tiende seconden uitlezen, ook al heeft u deze verder niet nodig.

Om de tijd in te stellen moet eerst bit 7 van adres 56335 worden uitgezet met POKE 56335,PEEK(56335) AND 127. Vervolgens moet bit 7 van adres 56334 worden aangezet met POKE 56334,PEEK(56334) OR 128. Dit bit geeft aan of de frequentie van het lichtnet 50 Hz (bit is aan) of 60 Hz (bit is uit) is. In Europa is die frequentie 50 Hz. Vervolgens kan in de tijdadressen de juiste tijd worden gepoked. U moet dan wel weer opletten dat de waarden BCD getallen zijn. Als het aantal minuten bijvoorbeeld 25 is moet in adres 56330 niet 25 maar 16*2+5 worden gepoked. Om dezelfde reden als bij het lezen wordt het veranderen van de adressen gestopt na het poken van de uren en weer gestart na het poken van de tiende seconden. Het volgende programma is een voorbeeld van de werking van de klok:

Adres	Functie
56328	Tiende seconden (BCD)
56329	Seconden (BCD)
56330	Minuten (BCD)
56331	Uren (BCD) bit 0-3: rechter cijfer bit 4 : linker cijfer bit 7 : 1=PM, 0=AM
56333	Alarm bit 2 : 1=alarmtijd bereikt
56334	Frequentie bit 7 : 1=50 Hz, 0=60 Hz
56335	Tijd zetten bit 7 : 0=zet gewone tijd 1=zet alarmtijd

Figuur 1. Adressen van de TOD klok

```
5 rem tod klok
10 print "voer tijd in:"
20 poke 56335,peek(56335) and
   127
25 poke 56334,peek(56334) or
   128
26 gosub 500:print chr$(147)
30 u1=peek(56331) and 15
40 u2=(peek(56331) and 16)/16
50 u$=chr$(u2+48)+chr$(u1+48)
60 m1=peek(56330) and 15
70 m2=(peek(56330) and
   240)/16
80 m$=chr$(m2+48)+chr$(m1+48)
90 s1=peek(56329) and 15
100 s2=(peek(56329) and
   240)/16
110 s$=chr$(s2+48)+chr$(s1+48)
120 a$="am"
130 if peek(56331) and 128
   then a$="pm"
140 print chr$(19)u$":
   "m$":"s$" "a$
150 ts=peek(56328):rem
   herstart klok
160 goto 30
500 input "uren";u
510 input "minuten";m
520 input "seconden";s
530 input "am/pm";a$
540 u1=int(u/10):u2=u-u1*10
550 poke 56331,u2+u1*16-
   (a$="pm")*128
570 m1=int(m/10):m2=m-m1*10
580 poke 56330,m2+m1*16
590 s1=int(s/10):s2=s-s1*10
600 poke 56329,s2+s1*16
610 poke 56328,0
620 return
```

Ook is het mogelijk om een alarmtijd in te stellen. Deze alarmtijd moet op dezelfde manier worden gezet als de gewone tijd; alleen moet van tevoren bit 7 van adres 56335 worden aangezet in plaats van uitgezet. Zodra de gewone tijd de alarmtijd heeft bereikt, zet de computer bit 2 van adres 56333 aan. Door vanuit een programma op dit bit te testen, kan een bepaalde actie worden uitgevoerd als de alarmtijd bereikt is (bijvoorbeeld een geluidssignaal geven). Dit testen kan als volgt gebeuren: IF PEEK(56333) AND 2^2 THEN actie. Nadat adres 56333 is uitgelezen, wordt het bit uitgezet.
In de computer bevindt zich nog een tweede TOD klok, waardoor het mogelijk is om tegelijk twee verschillende tijden bij te houden. Deze tweede klok heeft precies dezelfde faciliteiten als de eerste. Alle adressen die nodig zijn voor deze klok zijn 256 groter dan de adressen van de eerste klok. De tijdadressen bevinden zich bijvoorbeeld op 56584 tot 56587.

RAM onder $A000 en $E000

De Commodore 64 heeft een 6510 microprocessor. Deze microprocessor gebruikt 16 bits om een geheugenadres te representeren en derhalve kan de Commodore 64 niet meer dan 64 Kilobytes adresseren (Het grootste adres wat namelijk in 16 bits gerepresenteerd kan worden, is 65535). Deze 64K bestaan voor een deel uit RAM (Random Access Me-

mory; feitelijk is dit een slechte naam, want elk geheugen is een random access memory, ofwel willekeurig toegankelijk) en voor een deel uit ROM (Read Only Memory). In RAM kan gelezen en geschreven worden met PEEK en POKE, en in ROM kan alleen gelezen worden. Overigens is de SID (muziekchip) WOM, Write Only Memory. Hier kan dus alleen in geschreven worden. Bij het uitlezen krijgt u als waarden allemaal nullen. De Commodore bezit twee gebieden ROM elk ter grootte van 8K. Ten eerste het gebied van 40960 tot 49151 (hexadecimaal $A000 tot $BFFF). In dit gebied zit de Basic-interpreter. Dit gedeelte van de computer voert alle commando's en instructie's uit die u vanuit Basic invoert. Het tweede gebied loopt van 57344 tot 65535 ($E000 tot $FFFF) en in dit gebied zit het Operating System (de kernal) van de Commodore. Hier zit onder andere de communicatie met de randapparatuur. Verder zorgt de kernal ook voor het bijhouden van allerlei registers en adressen (bijvoorbeeld zero-page adressen).
Onder deze twee gebieden ROM zit een evengroot stuk RAM , dat u gewoon kunt gebruiken. Als u normaal vanuit BASIC een waarde ergens in dit gebied poked, dan wordt deze waarde in het RAM, wat verborgen zit onder het ROM, gepoked. POKE 41000,255 zet bijvoordbeeld waarde 255 in het RAM onder het Basic-ROM in adres 41000. Het is natuurlijk logisch dat deze waarde niet in het ROM geheugen wordt gezet, want in ROM kan alleen maar gelezen en niet geschreven worden. Doet u echter PRINT PEEK(41000) dan krijgt u wel de waarde van de betreffende ROM geheugenplaats. Op deze manier heeft u natuurlijk niets aan het RAM, want er kan zo wel in geschreven worden, maar niet in gelezen worden. Het is echter mogelijk om de ROM gebieden geheel uit te schakelen. De beide gebieden ROM kunnen onafhankelijk van elkaar worden uitgeschakeld. Het Basic-ROM wordt uitgeschakeld en vervangen door RAM door bit 0 van geheugenplaats 1 uit te zetten: POKE 1,PEEK(1) AND 254. Vanuit machinetaal kan dit natuurlijk gewoon gedaan worden, maar vanuit Basic slaat de computer natuurlijk goed vast als dit zonder meer gedaan wordt. Vanuit Basic heeft u immers de Basic-interpreter nodig. Als u toch vanuit Basic het RAM gebied wilt aanspreken, moet eerst het hele ROM gebied naar het RAM gekopieerd worden. Dit wordt als volgt gedaan:

```
FOR I = 40960 TO 49151:POKE
   I, PEEK(I):NEXT
```

Als dit gedaan is, kan het ROM worden uitgeschakeld met POKE 1,PEEK(1) AND 254. Nu kunt u bijvoorbeeld wat instructies van naam veranderen. Als voorbeeld de volgende poke: POKE 41853,63. Het woord READY zal dan veranderen. Er kunnen natuurlijk nog meer dergelijke grappen worden uitgevoerd, maar vanuit Basic heeft u al met al niet zoveel aan dit verborgen stuk RAM. Het Basic-ROM kan weer worden aangezet door bit 0 van adres 1 aan te zetten: POKE 1,PEEK(1) OR 1. In machinetaal heeft u wel veel aan dit onderliggende RAM, want bij het werken in machinetaal heeft u uiteraard geen Basic-interpreter nodig. Er kan bijvoorbeeld heel mooi een hires-plaatje van 8K in dit gebied worden opgeslagen.
Op dezelfde manier kan ook het tweede ROM gebied worden gebruikt ($E000 - $FFFF). Om dit gebied uit te schakelen moet het tweede bit van adres 1 op nul worden gezet: POKE 1,PEEK(1) AND 253. Vanuit Basic kan het RAM achter het ROM van de kernal niet worden aangesproken. In machinetaal kan dit wel worden gedaan. Als de kernal is uitgeschakeld heeft u echter geen ROM-routines meer (tenzij u natuurlijk een stuk kernal ROM naar het achterliggende RAM kopieert). Verder moet er ook heel goed worden uitgekeken met de interrupt. Alle handelingen van de interrupts (zoals IRQ-, break- en NMI-interrupt) worden namelijk uitgevoerd in de kernal. Daarom moet voor het uitzetten van de kernal, de interrupt uitgezet worden. Dit gebeurt met de machinetaal-instructie SEI (Set interrupt disable bit). Pas als u de kernal weer aanzet, mag de interrupt weer aangezet en gebruikt worden. Dit aanzetten van de interrupt gebeurt met CLI (Clear interrupt disable bit). Het volgende programma geeft een klein voorbeeld.

```
MNEMONICS       COMMENTAAR

sei              zet interrupt uit
lda $01         zet bit 2 van
and #$FD        adres 1 uit
sta $01
...              programma
lda $01         bit 2 van adres 1
ora #$02        weer aan
sta $01
cli              interrupt aan
```

BCD getallen

Een lastig probleem bij het programmeren in machinetaal is het printen van getallen. Het getal moet daarvoor namelijk ontbonden worden in eenheden, tientallen, honderdtallen etc., om te bepalen uit welke cijfers het getal is opgebouwd. Daarna moet van elk cijfer de ASCII-waarde bepaald worden. Zo kan het getal 25 worden ontbonden in 2*10 en 5*1. De cijfers waaruit 25 dus is opgebouwd, zijn dus 2 en 5. Deze cijfers kunnen worden omgezet in ASCII-waarden door bij elk cijfer de ASCII-waarde van het cijfer 0 op te tellen. (De ASCII-waarde van het cijfer 0 is 48.) De zo verkregen ASCII-waarden kunnen dan op het scherm worden gepoked. Het omrekenen naar ASCII is dus zeer makkelijk. De moeilijkheid zit hem echter in het ontbinden van het getal. Hiervoor moet namelijk gedeeld worden. Om bijvoorbeeld het aantal tientallen te berekenen in 25, moet 25 door 10 gedeeld worden en de uitkomst naar beneden

Het berekenen van ASCII-waarden voor BCD-getallen

MNEMONICS		COMMENTAAR
sed		decimale rekenmode
lda	#$15	laad accu met BCD 15
clc		
adc	#$10	tel BCD 10 bij accu
cld		normale rekenmode
pha		zet acxcu op stack
and	#$0f	bereken recht. cijfer
adc	#$30	bereken ASCII-waardde
sta	$0401	zet op scherm
pla		bereken linker cijfer
lsr		schuif accu 4 bits naar rechts
lsr		
lsr		
lsr		
adc	#$30	bereken ASCII-waarde
sta	$0400	zet op scherm

afgerond worden. In machinetaal is het uitvoeren van een deling echter een lastig karwei.
De Commodore 64 beschikt echter over een zogenaamde decimale rekenmode, waardoor het ontbinden van getallen een stuk eenvoudiger wordt. Normaal kan in 4 bits een getal tussen 0 (%0000) en 15 (%1111) worden gerepresenteerd. (De technische term voor een groep van 4 bits is een nibble. De getallen die beginnen met een '%' zijn binaire representaties.) In een byte (2 nibbles) kan dan maximaal 255 (%11111111) worden gerepresenteerd. In de decimale mode kan echter maar een cijfer tussen 0 en 9 (%1001) per nibble worden gerepresenteerd, waardoor in een byte maximaal 99 (%10011001) kan worden gerepresenteerd. Getallen die op deze manier gerepresenteerd worden, worden Binary Coded Decimal (BCD) getallen genoemd. Zoals u uit de binaire representatie van 99 kunt zien, bevat het rechter nibble (bit 0-3) van het byte het rechter cijfer van het getal en het linker nibble (bit 4-7) het linker cijfer. Dit heeft tot gevolg dat de cijfers waaruit een getal is opgebouwd, heel eenvoudig bepaald kunnen worden; door het byte te vergelijken door middel van AND met 15 (%00001111) wordt het rechter cijfer verkregen. Door te vergelijken door middel van AND met 240 (%11110000) en vervolgens de uitkomst 4 bits naar rechts te schuiven (met 4 keer LSR) wordt het linker cijfer van het getal verkregen. Deze cijfers kunnen dan weer, zoals eerder is uitgelegd, omgezet worden in ASCII-waarden.
Om een register te laden met een BCD getal kan handig gebruik worden gemaakt van hexadecimale getallen. De cijfers van een hexadecimaal getal komen precies overeen met de waarde van de nibbles. Het linker nibble van het hexadecimale getal $15 heeft bijvoorbeeld de waarde 1 en het rechter nibble 5. Deze waarden komen precies overeen met het BCD getal 15. Een register kan dus eenvoudig met een BCD getal worden geladen via een hexadecimaal getal. LDA #$15 zorgt er voor dat de accu het BCD getal 15 bevat.

Om twee BCD getallen op te tellen of af te trekken, moet de computer eerst in de decimale rekenmode gezet worden. Dit gebeurt met de machinetaal instructie SED. Met CLD wordt de computer weer in de normale rekenmode gezet. U moet opletten dat u bij het omzetten van cijfers in ASCII-waarden de computer in de normale rekenmode zet, omdat ASCII-waarden geen BCD getallen zijn.
Het volgende programma telt 10 bij 15 op en zet de uitkomst op het scherm.
Soms moet vanuit Basic met BCD getallen gewerkt worden, zoals in de tip over de TOD klok. BCD getallen kunnen in een adres worden gepoked door middel van POKE adres, 16*linkercijfer + rechtercijfer. Dus POKE 49152,16*9+5 zorgt ervoor dat adres 49152 BCD getal 95 bevat. Het rechter cijfer van een opgeslagen BCD getal kan worden berekend met PEEK(adres) AND 15. Het linker cijfer met (PEEK(adres) AND 240)/16.

Michel de Boer & Hylke Spranger

Basicminiatuurtjes

Een rubriek van Alex van Maarschalkerwaart

Hier volgen een aantal miniatuurtjes die Toine Ermes uit Asten ons toezond. Let op, 3 van deze programma's zijn in Simons Basic.

Lo-Res Cirkel

Het eerste programma vraagt om twee waardes in te voeren en tekent daarvan een Lo-Res Cirkel.

```
10 input"{SHIFT CLR}hoogte{SPACE}(
   )";h
20 input"breedte{SPACE}( )";b
30 print"{SHIFT CLR}":forom=0to2*step
   .05
40 x=int(20+sin(om)*b)
50 y=int(12-cos(om)*h)
60 poke1024+x+40*y,160:next
70 goto70
```

Pyramide Freaks

Het tweede programma dat Toine Ermes ons toestuurde heet Pyramide Freaks. Dit programa rekent de maten van een pyramide uit.

```
10 input"{SHIFT CLR}hoogte{SPACE}pyra
   mide{SPACE}(in{SPACE}cm.)";hp
20 basis=hp*1.57075:zijde=hp*1.4945
30 print:print"4{SPACE}gelijkbenige{S
   PACE}driehoeken{SPACE}waarvan"
40 print:print"de{SPACE}basis{SPACE}m
   oet{SPACE}zijn:";basis;"cm."
50 print"de{SPACE}beide{SPACE}zijden:
   {3xSPACE}";zijde;"cm."
60 getq$:ifq$=""then 60
70 run
```

3D Figuren

Met het volgende programma(in Simons Basic) kan je zelf, door veschillende waardes in te voeren, 3d figuren ontwerpen.

```
10 input"{SHIFT CLR}getal{SPACE}a,b,c
   ";a,b,c
20 HIRES0,3
30 forx=0toastep5
40 ARC160,100,0,360,c,b,x,1:next
50 forx=0tobstep5
60 ARC160,100,0,160,c,x,a,1:next
61 forx=0tocstep 5
65 ARC160,100,0,90,x,b,a,1:next
70 getq$:ifq$=""then 70
80 NRM:run
```

Klein Kunst

Het programma "Klein kunst" van Toine Ermes is ook in Simons Basic geschreven. Het is een soort Kaleidoscoop (zie woordeboek).

```
 10 HIRES0,3:x=160:y=100:k=160:l=100
 20 fort=1to500:x1=x:y1=y:k1=k:l1=l
 30 r=int(rnd(1)*8)+1
 40 ifr=1orr=3orr=5thenx=x+2
 50 ifr=3orr=6orr=7theny=y+2
 60 ifr=2orr=4orr=6thenx=x-2
 70 ifr=4orr=5orr=8theny=y-2
 80 ifx<160orx>310thenx=160
 90 ify<100ory>190theny=100
100 k=160-int(x-160):l=100-int(y-100): z=1
110 iftest(x,y)=1thenz=0
120 linex1,y1,x,y,z:linek1,y1,k,y,z
130 linek1,l1,k,l,z:linex1,l1,x,l,z
140 next t
150 getq$:ifq$=""then150
160 run
```

Kubus

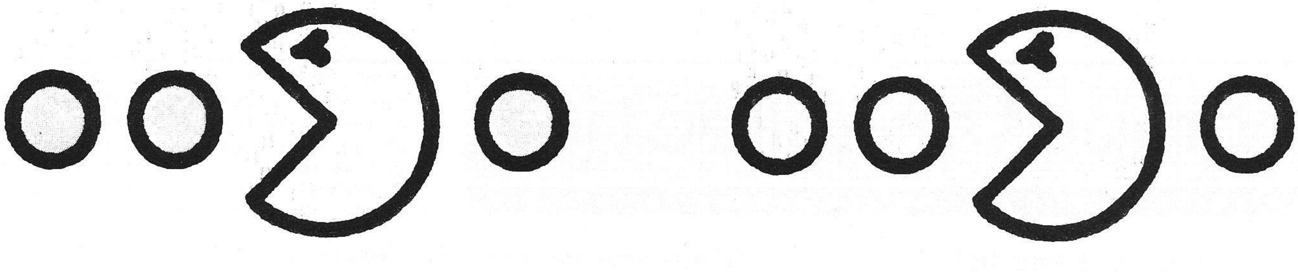
Het laatste programma dat Toine Erme ons toestuurde heet "Kubus". Dit programma is in Simons Basic geschreven en tekent verschillende soorten Kubussen op het beeldscherm.

```
10 HIRES0,3:fort=1to10
20 a=int(rnd(1)*40)+10:b=int(a/2)
30 x=int(rnd(1)*240)+1:y=int(rnd(1)*1
   10)+1
40 RECx,y,a,a,1:x1=x+a:y1=y+a
50 LINEx,y,x+b,y+b,1:LINEx+a,y,x+a+b,y+b,
   1
60 LINEx,y+a,x+b,y+b+a,1:LINEx+a,y+a,x+a+
   b,y+a+b,1
70 RECx+b,y+b,a,a,1
80 next
90 PAUSE5:run
```

Knipper

Laurens van Klaveren uit Harderwijk zond ons het programma "Knipper" toe, het geeft een leuk effect.

```
 10 a=49151
 20 a=a+1:readb:ifb=-1thensys49152:pok
    e53280,1:poke646,1:printchr$(147):
    end
 30 pokea,b:goto20
100 data120,174,20,3,172,21,3,142,25,1
    92,140,26,192,162,27,160,192,142,2
    0,3,140
110 data21,3,88,96,0,0
120 data173,32,208,141,33,208,173,134,
    02,101,01,141,33,208
140 data108,25,192,-1
```



Spiegelen

Dit programma is met recht een miniatuur, het werd ons toegezonden door Niels van Oosterrijk uit Hooglanderveen.

```
10 inputx:fort=7to0step-1:q=7-t:a=(x
   =2^t):x=x+a*2^t:s=s-a*2^q:next:pri
   nt s
```

Scroller

Dit programma van Martin Vermaat uit Hengelo zorgt ervoor dat het beeld scrold. (SYS 59761 ↓ en SYS 59765 ↑)

```
10 poke1844+x,30:poke56116+x,1:a=int(
   rnd(1)*40):poke1064+a,8:sys59761:p
   rint"{2xCRSR-UP}"
20 poke55376+a,0:get b$:ifb$="1"thenx
   =x-1
30 ifb$="7"thenx=x+1
40 if peek(1844+x)=81 then print"scor
   e";l:end
50 l=l+1:goto 10
```

Geluid text

Dit programma geeft een combinatie van tekst en geluid weer. Het is geschreven door Marnix van Asselt uit West-Terschelling.

```
 10 s=54272:poke s+24,15:pokes+5,0:pok
    e s+6,247
 20 print"{SHIFT CLR}":poke 53281,0:po
    ke 53280,0:poke 646,1
 30 a$="commodore{SPACE}info"
 40 forx=14 to 1 step-1
 50 poke 214,11:print
 60 b$=mid$(a$,x,1)
 70 if b$="{SPACE}"then nextx
 80 fory=1to12+x:printtab(y)"{CRSR-LEF
    T}{SPACE}";b$;:forz=1to8:nextz,y:g
    osub 100:nextx
 90 goto90
100 poke s+4,17:poke s,0:poke s+1,12:p
    oke s+4,16:return
```

Afgrond

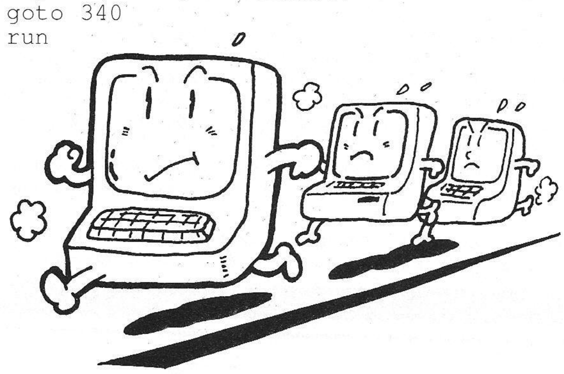
Dit programma komt ook van Marnix van Asselt en is een kleine demo van een bal.

```
 10 s=54272:pokes+24,15:pokes+5,o:poke
    s+6,248
 20 print"{SHIFT CLR}"
 30 poke 214,17:print:print"{11xCOM Y}
    M":printtab(16)"{14xCOM Y}"
 40 fora=1to3
 50 readw,x,y,z
 60 forb=xtoy
 70 poke214,w+(b^2)/4:poke211,b+z:prin
    t"{CRSR-LEFT}{SPACE}"
 80 poke214,w+(b^2)/4:poke211,b+z:prin
    t"Q{SPACE}"
 85 ifz=27thennextb:goto100
 90 nextb:gosub160:nexta
100 print"{CRSR-LEFT}{SPACE}"
110 poke214,15:poke211,15:print"aaaarg
    hhh!!!"
115 pokes+6,249
120 fora=90to45 step-.4:pokes+4,17::po
    kes,0:pokes+1,a:pokes+4,16:next
130 forg=1to100:next
140 pokes+13,250:pokes+8,2:pokes+11,12
    9:pokes+11,128
150 end
160 pokes,0:pokes+1,12:pokes+4,17:poke
    s+4,16:return
170 data 0,0,8,0,5,-6,7,15,10,-4,7,27
180 goto50
```

Snake

Bij dit programma (van Stevan Vijfwinkel uit Zwijndrecht) moet je er voor zorgen dat met slang tussen de punten stuurt. Het programma werkt met de joystick.

```
 10 poke53281,0:poke53280,0:print"{SHI
    FT CLR}{CTRL 6}"
170 for z=1to3
180 printchr$(147):ti$="000000":q=1444
    :pokeq,81
190 print"programmed{SPACE}by:t.g.t{SP
    ACE}en{SPACE}t.m.t{SPACE}on{SPACE}
    1-4-'89"
200 pokeq,81:j=peek(56320):j=15-(j and 15)
210 if j=4 then q=q-1:poke q,81:pokeq+
    1,32
220 if j=8thenq=q+1:pokeq,81:pokeq-1,3
    2
230 a=int(0+(39)*rnd(1))
240 printspc(a)"{CTRL 8}.{CTRL 6}"
250 if peek(q+40)32 then280
260 goto 200
280 v=54296:w=54276:a=54277:h=54273:l=
   54272
290 for x=15to0step-1:pokev,x:pokew,12
    9:pokea,15:pokeh,40:pokel,200:next
300 pokew,0:pokea,0
320 printchr$(147)"{6xSPACE}u{SPACE}he
    eft{SPACE}het"int(ti/60)"seconden{
    SPACE}volgehouden"
330 print"{7xSPACE}press{SPACE}fire...
    "
340 ifpeek(56320)=111then360
350 goto 340
360 run
```

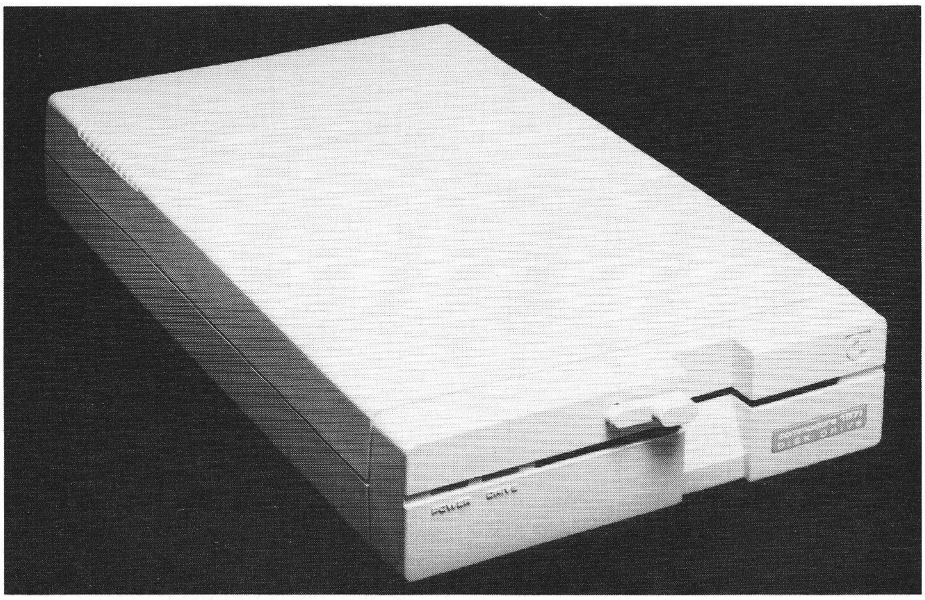


Veel lezers van ons blad sturen ons een briefje met daarin vragen omtrent de 1581 van Commodore. Aangezien wij zeer gevoelig zijn voor dit soort vragen, omdat wij zelf ook graag weten wat er op de markt verkrijgbaar is en willen weten hoe dat dan werkt, komt hier ons verslag.

De 1581 diskdrive

Het beschrijven van een drive is niet iets wat je nu vaak zal tegenkomen. Maar omdat de 1581 een zeer populaire drive blijkt te zijn, lijkt een nadere bestudering op zijn plaats. Zeker gezien het feit dat dit momenteel de enige losse drive is die verkrijgbaar is voor de C128.

Zoals we allemaal nog wel zullen weten, is er al een hele reeks van verschillende typen diskdrives van Commodore verschenen. Een heel klein kastje waarin het loopwerk zit, en een aparte voeding, is het laatste model van Commodore. Doordat de voeding nu in een apart kastje zit zult u niet zo snel hitte problemen krijgen met de 1581. Nee het is geen broodrooster en ook geen tosti-ijzer meer. We kennen allemaal de 1540 en 1541 toch nog wel. Geruime tijd later, met de introductie van de Commodore 128, kwamen daar de 1570 en 1571 modellen bij. Ook in Amerika verscheen er een dubbele diskdrive, model 1572. De gehele Commodore 128 KERNAL is hiervoor ontwikkeld. Maar wij in Nederland hebben hiermee nooit te maken gehad. Dat wil zeggen, met het dubbele model van de 1571.



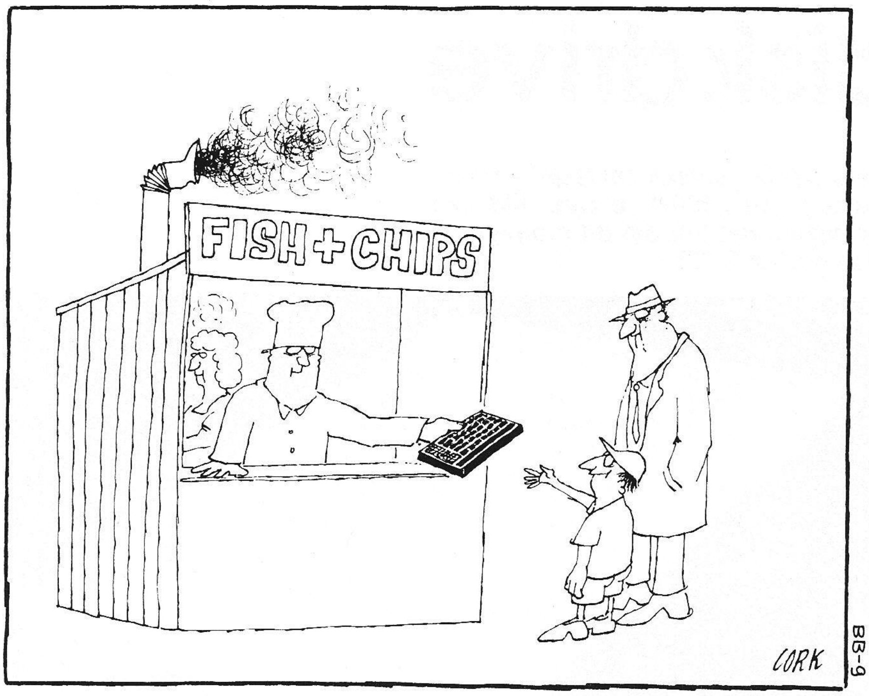
De veel bekendere Commodore 1571 Diskdrive

Commodore 64 en 128

U kan met uw C-64 de 1570 en 1571 best gebruiken. Maar hierdoor is er nog geen sprake van snelheidswinst. Die snelheidswinst is wel waarneembaar bij de C-128, waar de drive eigenlijk voor is ontworpen. Maar Commodore wil de C-64 gebruiker blijkbaar ook van het grote data opslag voordeel laten profiteren. Helaas beschikt de C-64 niet over de snelle seriële bus die bij de C-128 voor de snelheidswinst zorgt. Maar een aantal handige personen wist hier wel weer een mouw aan te passen. Alleen jammer is het, dat met die aanpassing dan de diskdrive niet helemaal compatible meer is. Zo klapt de software die met een disk-protectie wordt geleverd, heel vaak op z'n gat. Geen beginnen aan en maar weer terug naar de langzame mode van de C-64, terug naar af. Of bent u daar niet zo weg van en gaat u op zoek naar personen die voor u de disk-protectie verwijderen. Tja, daar worden wij dus even stil van. Deze personen zult u straks nog wel eens nodig kunnen hebben. Er is bijna nog geen software verkrijgbaar die direct zonder problemen met deze diskdrive wil samenwerken. Dat wil zeggen, de beschermde software met de zogenaamde disk-protectie. Ook GEOS werkt met behulp van de 1581 perfect, alleen is de opstart-disk niet op dit formaat verkrijgbaar. En het kan misschien ook nog wel even duren voordat daar verandering in komt. Tot die tijd zult u met de oude vertrouwde 1570 of 1571 GEOS moeten opstarten. Maar het wegschrijven van uw data gaat met GEOS prima richting 1581; En dat is tenminste iets. Zo wil Superbase helaas niet zijn data op de 3.5 inch diskette van de 1581 wegschrijven. Superbase laat het afweten wegens het niet kunnen samenwerken met de nieuwe DOS van de 1581. Over het hoe en waarom moeten wij u op dit ogenblik helaas het antwoord echter nog schuldig blijven. Dat gaat natuurlijk ook geen eeuwen meer duren. U krijgt in ieder geval de melding 70, No Channel, op uw beeldscherm te zien. Ook al zijn er Channels voldoende beschikbaar, Superbase verrekt het gewoonweg. Hoogstwaarschijnlijk gebruikt Superbase een channel dat niet meer, of nu voor andere doelen wordt gebruikt. En laten we eerlijk zijn, leuk is het in ieder geval niet. Mocht u zelf de oplossing hebben gevonden, klim dan direct in de pen. U weet niet half hoeveel personen om de juiste oplossing zitten te springen. Deze eer is dus aan de welwillende INFO-lezer! Dus voor de door u zelf geschreven software moet het al helemaal geen probleem zijn om op disk te kunnen lezen of schrijven. Uw software gaat vaak toch niet zo diep het DOS in als bij de grote firma's het geval is; En die doen dit ook alleen om er voor te proberen te zor-

gen dat u de diskette niet kunt kopiëren. Lukt u het wel dan heeft u geluk. De gewone save en load commando's werken dus prima met de 1581 samen. Maar dieper het DOS in gaan zal problemen met zich kunnen meebrengen.

weer tegen. Eén ding is zeker, de 3.5 inch diskettes zijn een stuk beter tegen het 'per ongeluk' verkeerd vastpakken beschermd. Lekkere vette vingers kunnen nu gelukkig een stuk minder snel voor data-verlies zorgen. Oké, het is nog mogelijk, maar dan



heeft u wel hele vreemde kapriolen moeten uithalen. Om te voorkomen dat u al weer 'per-ongeluk' een diskette gaat formatteren, is ook iets heel handigs uitgevonden. Nee, niet door Commodore maar door de diskette fabrikanten. Nu is het geen sticker-rommel meer, die het perongeluk formatteren moet tegen gaan, maar een heel handig plastic schuifje. Handig als wat, maar dan zult u wel even goed moeten opletten bij het bewerken van het schuifje. Bij de 5.25 inch diskette kon u het uitgeknipte gedeelte van de diskette afplakken, en klaar is Kees. Maar nu werkt dat precies andersom. Schuifje dicht is 'schrijven maar', en schuifje open is 'schrijven vergeten maar'. Tja opletten dus en niet per ongeluk de boel in de war schoppen met bijvoorbeeld de header (formatteer)-opdracht.

De interne techniek

Dat de 1581 technisch geheel volgens een nieuw concept is ontwikkeld kunt u natuurlijk wel op uw vingers natellen. In plaats van de O zo veel gebruikte 5.25 inch diskette wordt er nu van een 3.5 inch diskette gebruik gemaakt. Dat is dus al voordeel nummer een. Waarom? Omdat op dit soort diskettes veel meer data opslag mogelijk is. En daar draait het bij diskettes toch allemaal om. Konden we eerst met behulp van de 1571 340 Kilo byte data kwijt, nu kunnen we 800 Kilo byte op de 3.5 inch diskette van de 1581 wegschrijven. Iets meer dan u tot op heden met uw oude vertrouwde 1541 of 1571 gewend was. In totaal kunt u nu dus zo'n slordige 460 Kilo byte meer op diskette kwijt. De 1571 kost ongeveer zo'n *f* 700,= en dat kost een 1581 ook ongeveer. Gezien het grote opslag-voordeel is de prijs dus ook gunstig te noemen. Maar omdat de 3.5 inch diskettes ook duurder zijn, moet u dus niet al te vroeg gaan zitten juichen. Dat doen de diskette verkopers wel voor u, zullen we maar zeggen. Ook is dit maar een grapje, natuurlijk, want dan is de aanschaf voor de dealer, niet te verwarren met..., ook een stuk duurder en op deze manier valt zijn winst dan ook

Programmeren

Heel handig, denken sommige mensen, op diskettes te kunnen besparen door geen dubbele, maar enkelzijdige diskettes te kopen. Vaak gaat het goed, maar eens gaat het fout. Wij geven onze software in ieder geval niet de kans om op die manier te verdwijnen. Geen sprake van! O.k., het kost wel meer geld maar daar is de kwaliteit dan ook naar. Wij moeten er dus niet aan denken het vele 'bloed zweet en tranen' in rook te moeten zien opgaan. Nee wij kijken wel lekker uit, u gaat u gang maar. U weet het, 'Een gewaarschuwd mens telt voor twee'. Het zal dus ook niet aan de verkeerde voorlichting kunnen liggen. U heeft geen geld! Tja, dat is jammer maar ook daar hebben wij een goed antwoord op. Een beperkt antwoord weliswaar, maar toch. Kruip achter de computer en programmeer er op los. Maak er wat van en dan kunt u ook nog eens wat bijverdienen! En daar schaft u dan gewoon uw computerspullen van aan. En vergeet het niet, opsturen naar COMMODORE INFO die hele hap. Wij helpen u, u helpt ons. Ietwat filosofisch misschien, maar op die manier blijft onze dobber ook lekker boven water. En wordt de dobber zwaarder, dan geeft u er gewoon weer een ruk aan! U bent degene die de programmeerteugels in de hand heeft, niet wij. Kortom, wij zien uw werk wel verschijnen. Maar goed terug naar waar wij gebleven waren. De 15XX drives hadden 'maar' 2Kb intern RAM terbeschikking. En daarmee kon niet een gehele track worden opgeslagen in RAM. Maar nu wel omdat er 6Kb RAM is bijgekomen. Samen wordt dat dus 8Kb RAM geheugen voor de 1581. Met een heel handige routine die 'Track Cache Buffer Loader' heet, wordt het RAM met de data van de disk volgepropt. Op deze manier gaat er nu een hele track met data richting Cache-buffer. Eerst moest u die sector van diskette inlezen, maar nu wordt die data uit het interne RAM opgehaald. Vandaar dat de 1581 ook iets sneller werkt. Jammer genoeg is dit maar ongeveer 10%, maar alles is mooi meegenomen. De remmende faktor is dus nog steeds de seriële bus van Commodore. Maar door deze truc is het data ophalen wel mooi iets sneller geworden. Dit komt doordat RAM to RAM copy's veel sneller zijn, omdat niet eerst de disk zijn diverse electronica-besturingen behoeft te activeren. Dus geen motor aansturen, die de lees/schrijfkop heen en weer stuurt, en dat soort dingen meer.

Stappenmotor

Ook zijn enige komponenten vervangen door andere typen componenten. Zo is de 6522 vervangen door een 8520. Een hele dure jongen die ook in de Amiga voorkomt. Eén keer de printer aansluiten bij de Amiga, is vaak al voldoende om dit IC om zeep te helpen. En dat moest natuurlijk wel, om-

dat de logische schakelingen in de diskdrive geheel ander werk moeten verrichten. De stappenmotor, de motor die de bewegingen van de lees/schrijfkop regelt, moet nu bijvoorbeeld *ook naar track 80 kunnen* springen. De in de 1570 en 1571 gebruikte WD 177X doet nu tenminste wat hij moet doen. De in de 1570 en 1571 gebruikte controller van Western Digital bestuurde de stappenmotor niet. Nu is dat wel het geval en vandaar dat dit ook een stuk sneller gebeurt. De signalen voor write-protect en track 0 detectie stonden ook op deze controller om te voorkomen dat er per ongeluk op een disk wordt geschreven die afgeplakt, en dus beveiligd zou moeten zijn. Zo kan een diskcommando als COLLECT wat gewoonlijk uren schijnt te duren, nu in enkele seconden zijn werk verrichten op de 3.5 inch diskette van de 1581. Ook hier weer winst in snelheid.

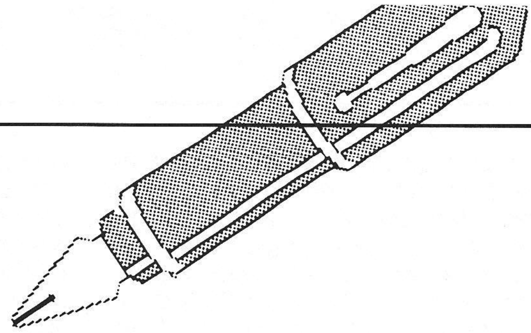
Massa opslag

En wij vonden het al een hele verademing. 340Kb opslag met een 1571 was al een hele verbetering ten opzichte van de 1541 en 1570 van Commodore. Maar nu kunt u geen 1328 blokken maar liefst 3160 blokken wegschrijven naar disk. Dat is dus 2.4 keer zoveel als op een 1571 kan worden weggeschreven. Met zo'n grote opslagcapaciteit zult u meer files wegschrijven dan op een normale 5.25 schijf. Op de 3.5 inch schijf is het *dan ook mogelijk om inplaats van* 144 files, 296 files weg te kunnen schrijven. Dit is dan dus een winst van 152 files. En wat dacht u van een SUBDIRECTORY op disk? Ja, dat is met zo'n 1581 ook mogelijk geworden. O.k., zo'n sub-directory slikt wel even 40 blokken opslag voor de BAM, Block Availability Map. Elke directory neemt dus 40 blokken in beslag voor zijn directory en BAM. In de directory staan de wijzers naar de files die u eventueel op disk heeft staan. Ook staat daar het type file enz. Met andere woorden, wanneer u een sub-directory aanmaakt van 200 blokken, dan mist u er geen 200 maar 240! Een kwestie van in het achterhoofd houden. De hoofd directory wordt ROOT directory genoemd. Om naar een andere directory te kunnen overswitchen zult u wel eerst naar de ROOT terug moeten. Vanuit de ROOT kunt u dan naar een bestaande sub-directory overswitsen. Met behulp van zulke sub-directory's wordt het werken op disk een stuk moeilijker dan u misschien wel denkt. Er zijn vrij ingewikkelde opdrachten voor nodig om een sub-directory te kunnen aanmaken. U kunt dit zien in het door Henk-Johan van Rantwijk gemaakte programma, dat u in dit nummer kunt vinden. Wie zo'n 1581 eenmaal heeft zal daar in *ieder geval veel plezier van kunnen* hebben. Ook heeft hij een soort WORKBENCH geschreven voor deze diskdrive, die wij later ook zullen publiceren in de INFO.

Conclusie

Tot slot kunnen we de lezer er op attenderen dat wij zeer te spreken zijn over de 1581, zeker gezien het feit dat er nu met sub-directory's gewerkt kan worden. Ook de massale data opslag is gewoonweg een verademing. De bijgeleverde demo-diskette is zeker voor in het begin heel handig. Als advies aan de aspirant koper moeten we dit nog een keer kwijt: koop alleen dubbel-zijdige diskette's voor deze diskdrive. De verkoop prijs van zo'n *f* 700,= is, zeker gezien de mechanische en electronische afwerking, niet te duur. Zelf vinden wij het voor een hobby drive wel wat te duur. Wij hopen dat u de 1581 bij uw hardware leverancier kunt verkrijgen. Zeker is dat u wel even moet zoeken, voordat u de 1581 kunt aanschaffen.

De populaire GEOS vraag- en antwoord-rubriek.



GEOS-INFO

Alweer een printerprobleem

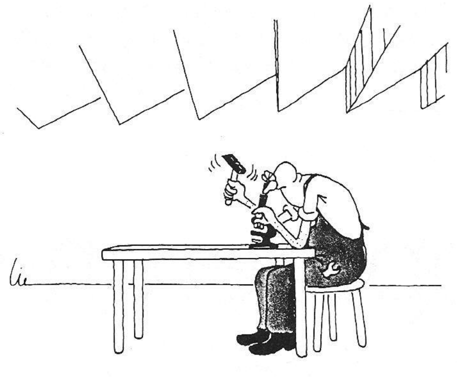
De heer van Elten uit Ede kan zijn printer, een Commodore 1525, niet aan de praat krijgen met de GEOS 1.3 versie. Zoals reeds in vorige afleveringen van deze rubriek is beschreven, dient u in zo'n geval een andere printer-driver te proberen. Vaak worden de oorspronkelijke printermogelijkheden aangestuurd door een andere printerdriver. In dit geval dient u de printer-driver MPS 801 te gebruiken.

Plaatjes in GeoWrite

Uit Leidschendam ontvingen wij een brief van dhr. van Loon, met de vraag waarom een tekening, die uit twee delen bestaat niet precies op elkaar te passen is. Hij heeft namelijk een kruiswoordpuzzle in GeoPaint getekend, welke bestaat uit twee delen, en vervolgens naar de foto-manager gecopieerd. Daarna heeft dhr. van Loon de twee plaatjes in een GeoWrite document ingelezen, in de hoop dat deze precies boven elkaar geplakt konden worden. Dit is niet gelukt. In de GEOS 2.0 versie is het mogelijk om plaatjes tot op de pixel nauwkeurig op elkaar aan te laten sluiten. In dit specifieke geval is het misschien verstandiger om de tekening gewoon in GeoPaint te laten staan en de tekst vanuit GeoWrite, door tussenkomst van de text-manager, te copiëren naar hetzelfde GeoPaint document. Bij gebruik van de 1.3 versie van GEOS is het niet mogelijk om een tekening, anders dan in het midden van de pagina, te positioneren. Tevens is het niet mogelijk om tekst naast een tekening in te voeren. De versie 2.0 kent deze mogelijkheden echter wel. In beide versies is het onmogelijk om een hele pagina uit GeoPaint in een foto-album te plaatsen, alleen tekeningen welke maximaal de grootte van het tekenvenster hebben kunnen hierin geplaatst worden.

Positief en wederom de REU.

Van de heren Joost en Frans Halenbeek uit Zeist ontvingen we een nuttige recatie op het probleem betreffende de aanschaf van de **REU** (Ram Expansie Unit). Ze waren zelf inmiddels tot de ontdekking gekomen dat een dergelijke REU in Nederland niet of nauwelijks verkrijgbaar is. Zijzelf hebben hun REU (1764) uit Duistland gehaald. Hierbij worden plaatsen als Essen, DÜsseldorf of Aken genoemd, waarbij in de zaken als Quelle of Kaufhoff de drie gangbare modellen verkrijgbaar zijn. De 1764 komt dan op DM 299,00 en de 1750 op DM 199,00. Verder weten ze nog te melden dat de Font-editor uit het boek "GEOS Tips & Trucs" van Data-Becker alleen werkt met de versie 1.2 van GEOS. Zij adviseren dan ook om FontPack Plus aan te schaffen en zodoende gelijktijdig in het bezit te komen van de GEOS Font-Editor. Hartelijk dank voor jullie positieve bijdrage.



Vliegende pointer

Het aanwijspijltje (pointer) van de GEOS 1.3 van de dhr. G.Z. Kamerling uit 2e Exloërmond vliegt steeds naar de rechterkant van het scherm. Bij een kennis van hem doet zijn GEOS dit echter niet. Vreemd genoeg dus. Als het pakket op de ene computer goed functioneert, zal het naar alle waarschijnlijkheid toch aan uw apparatuur liggen. Misschien is uw joystick niet geheel in orde. Voor ons is uw probleem echter een raadsel, waar we geen pasklare opslossing voor hebben.

GEOS en de MPS 1230

Het probleem dat dhr. F.J. de Weger heeft met zijn nieuwe Commodore MPS 1230 printer en GEOS 2.0 ligt waarschijnlijk in het feit dat deze printer betrekkelijk nieuw is. Berkeley Softworks heeft de GEOS 2.0 versie al in juli van het vorige jaar gecompleteerd en was kennelijk niet in staat op de nieuwste ontwikkelingen op de hardware markt in te spelen. Het printen in NLQ mode met GEOS zal voor u betekenen dat u de dipswitch instelling aan moet passen. Op uw vraag of het mogelijk de DATE functie, bij gebruik van headers en footers, in het Nederlands te laten afdrukken, moet ik u helaas teleurstellen. GEOS is en blijft een Amerikaans produkt en zoals u wellicht weet werken de Amerikanen met een omgekeerd datumstelsel. Misschien als er nog eens een Nederlandse versie van GEOS 2.0 op de markt komt, dat dit probleem dan tot het verleden hoort.

Alweer printerproblemen...

Al enkele malen eerder hebben wij gemeld dat het haast ondoenlijk is om specifieke printerproblemen, welke voortvloeien uit niet al te gebruikelijke hardware samenstellingen, op te lossen. Dit geldt dus ook voor de heren Bockx uit Kessel (België) en Th. Neve uit Kloosterzande. De minste problemen heeft u wanneer u een in de handleiding beschreven configuratie gebruikt. In het laatste geval wordt ook nog eens een diskdrive (OC-118 N) gebruikt, hetgeen onherroepelijk tot problemen kan leiden. Helaas kunnen we hiervoor geen oplossing aandragen.

GEOS gebruikersgroep

Laatst ontvingen we uitdraai van iemand, bestaande uit zo'n slordige 8 pagina's. Op elke pagina stond wel een vraag, die verwees naar de afgedrukte tekst. U zult begrijpen dat het niet doenlijk is om op zulke uitgebreide problemen in te gaan. Dit soort problemen horen meer thuis bij een gebruikersclub. Sinds kort kennen we in Nederland zo'n club, die is gevestigd in Almere. De Geos Gebruikers Groep, telefoon 03240-10041. Hier kunt u met al uw terecht voor uw aanloopproblemen en gebruikersvragen.

Zoals in de laatste aflevering van deze cursus (nr. 2/'89) werd beloofd, krijgt U deze maand de listing van "harcopy.prg", een programma waarmee beeldschermuitdraaien van GEOS kunt maken. Aangezien in dit programma vrij moeilijke machinetaaltechnieken aan de orde komen, zal Peter Boncz deze keer ook enige uitleg aan pure machinetaal wijden.

GEOS-machinetaal (5)

Hardcopies maken

De printerdrivertechniek, waarover de vorige aflevering van deze serie ging, zal in de listing van deze keer gebruikt worden. Om het geheugen nog eens op te frissen: vorige keer werd het programmaatje "hardcopy" gelist. Deze keer krijgt U "hardcopy.prg" dat daarmee samenwerkt. Het programma van vorige keer is altijd hetgene dat U opstart als U een hardcopy wilt maken. Het scherm wordt dan wit, dit betekent dat als U op CTRL-H drukt, de hardcopy uit de printer zal komen rollen. De applicatie waaruit "hardcopy" opgestart werd, loopt tussen dat moment en het indrukken van de CTRL-H, nog gewoon door.

Pas op het moment dat op CTRL-H gedrukt wordt, zal het programma waarin de echte print-techniek verwerkt zit van de disk worden gehaald: "hardcopy.prg". Let dus wel op dat BEIDE programma's altijd op disk staan.
Het programma "harcopy.prg" print hardcopies die zowel in de breedte als in de hoogte verdubbeld zijn. Hierom is het ook nodig, dat de hardcopy op z'n kant uitgeprint wordt. Anders zou hij immers te breed worden voor sommige printers. Kromme printers als de Brother HR5 en de MPS 801, die maar 480 puntjes in de breedte hebben (i.p.v. 640), kunnen nu met het programma werken. De afmetingen van de hardcopy nu zijn immers 2x200 (breedte) bij 2*320 (lengte).

Dit kantelen en verdubbelen betekent wel dat er nogal wat gerekend zal moeten worden, en dat in machinetaal. De C-64 heeft al zo weinig machinetaalinstructies, dat men hem wel grappend de eerste Reduced Instruction Set Code (RISC) machine noemt. In dit soort gevallen is dat niet RISC, maar gewoon CRISis. Vermenigvuldigen zal bijvoorbeeld met zogenaamde schuifoperaties in combinatie met optelinstructies moeten gebeuren. Voorbeeldje: vermenigvuldigen met 9 wordt dan opgesplitst in driemaal met 2 vermenigvuldigen en eenmaal optellen. (2*2*2+1=9).



Algemene machineopbouw

Hoewel een groot gedeelte van wat ik nu ga bespreken eigenlijk in deze serie bekend verondersteld wordt, zal ik vanwege de moeilijkheidsgraad van het hardcopyprogramma toch bij het begin beginnen. Het hart van de computer wordt gevormd door de CPU, grofweg aan te duiden met 'processor'. In het geval van de C-64 is dat de 6502 chip. Een zogenaamde bus verbindt deze CPU met zaken als het geheugen, het beeldscherm, het toetsenbord, diskdrives en printers. De CPU is het hart van de computer, dat zich constant bezighoudt met het uit het geheugen halen van instructies om die vervolgens uit te voeren. De CPU van de C-64 heeft 3 registers: a,x en y. Registers zijn interne variabelen in de processor, die veelvuldig gebruikt worden, en noodzakelijk zijn voor het programmeren. Al deze drie registers hebben de grootte van 1 byte in de C-64. Andere -modernere- processors hebben veel meer registers, de MC 68000 heeft er bijvoorbeeld 16, en dan nog van 32 i.p.v. 8 bits. Naast de 3 algemene registers heeft de C-64 een aantal statusbits. Statusbits bevatten een 1 of een nul, ofwel ze zijn 'gezet' of niet. Er is bijvoorbeeld een N statusbit, dat gezet wordt door de processor als een register een negatieve waarde krijgt. Er is een Z statusbit, dat gezet is als bij een vergelijking (cmp/cpx/cpy) gelijk wordt geconstateerd. Tevens is er het zogenaamde Carrybit, dat aanstaat als een register de waarde 255 overstijgt.

Schuifoperaties

Dit Carrybit speelt een belangrijke rol in de twee schuifinstructies die in de listing gebruikt worden. Deze twee zijn ASL en ROL, ofwel Arithmetic Shift Left, en ROtation Left. In schema 1 vindt U een overzicht van beide schuifoperaties. Zoals de naam al veronderstelt, schuiven ze beide de bits naar links in een byte. Verder wordt bij allebei de inhoud van de vroegere zevende bit (een byte heeft 8 bits: bit 0 tot 7) in de Carry (aangeduid met C) geplaatst. Het verschil tussen de twee schuifoperaties zit in de waarde van de nieuwe 0-de bit. Bij ASL wordt deze 0-de bit altijd leeg, bij ROL krijgt de 0-de bit de vroegere waarde van de Carry.

ASL

Het effect van ASL is een zuivere vermenigvuldiging met twee. Als op een byte met waarde 1 (dus de nul-de bit aan, de andere bits uit) ASL wordt losgelaten, dan is het resultaat een byte met waarde 2 (de eerste bit aan, en de rest uit). De Carry is nul, want dit is de waarde van de (vroegere) zevende bit. Maar wat is er nu precies aan de hand als de Carry na afloop wel aan is? Daarvoor bekijken we het geval van een byte met bit zeven aan, dus waarde 128. Na ASL zijn alle bits uit, en bit 7 is in de carry geschoven. De waarde van de byte is dus nul geworden, terwijl hij eigenlijk 256 had moeten zijn. Het is dus duidelijk dat de Carry aangeeft of er OVERFLOW is opgetreden. Het getal is te groot geworden voor een byte, en zou dus eigenlijk in twee bytes moeten staan.

ROL

Voor het verdubbellen van getallen die in twee bytes staan opgeslagen (een Low- en een High-Byte) volge men dan het volgende recept: bewerk de LowByte met een ASL. De LowByte is dan verdubbeld, en als er overflow was, dan is de Carry 1, anders is de Carry 0. Vervolgens wordt de HigByte geROLd. De HighByte wordt dan ook verdubbeld, en de Carry wordt in de 0-de bit geplaatst. Het uiteindelijke effect is een kloppende verdubbeling, want in geval van overflow van de LowByte wordt dit gecompenseerd, doordat de HighByte een extra hoger wordt.

Optellen en aftrekken

Optellen gaat vrij gemakkelijk op de C-64. Daarvoor is de instructie ADC bestemd. Deze heeft het volgende effect:

Accu := Accu + Waarde Adres + Carry
(":=" betekent "wordt")

Wil men bijvoorbeeld bij waarde van adres $1000 tien (hex. $0a) optellen, dan moet dat zo gebeuren:

```
clc          ; Carry := 0
lda #$0a     ; Accu := 10
adc $1000    ; Accu := 10 +
    waarde $1000 + 0
sta $1000    ; $1000 := Accu
```

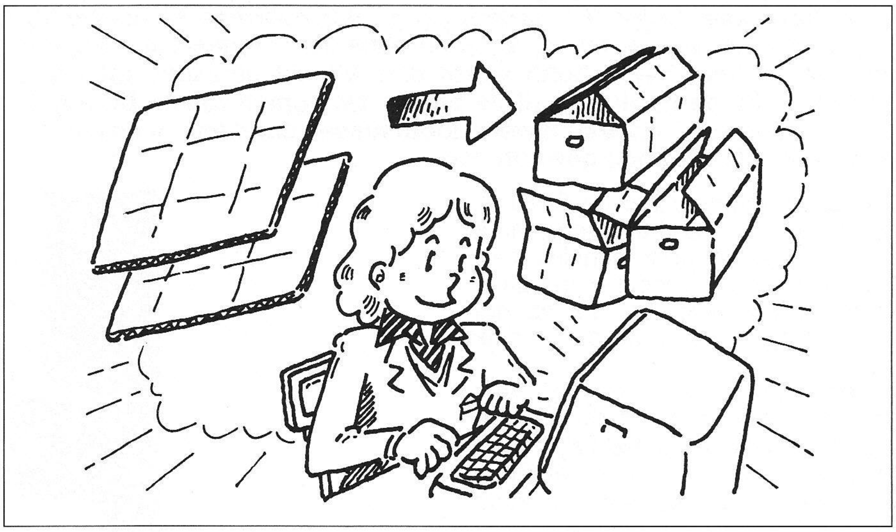
En na afloop bevat de Carry een 1 als er overflow is opgetreden, en een nul als de optelling bij elkaar minder dan 256 was. Parallel hieraan ligt de instructie SBC, met de volgende functie:

Accu := Accu - Waarde Adres - Carry.

FillBuffer

De horizontale rij van 80 naast elkaar liggende CARDS (blokjes van 8x8) die FillBuffer op een of andere manier uit de rij van 4 breed zal moeten destilleren, wordt per CARD bepaalt. Uiteindelijk zal FillBuffer echter niet 80 CARDS, maar slechts 50 moeten maken. De andere 30 zijn gewoon leeg. (dit komt omdat het scherm oorspronkijk 200 bits hoog is, gekanteld en verdubbeld worden dit dus 400 bits breed, en dat zijn precies die 50 blokjes van 8 breed)

De vertikale rij wordt dus gesplitst in 50 blokjes van 4 bij 4. De procedure "Haalvierbijvier" kantelt zo'n blokje en maakt er 8 bytes van (lengte- en breedteverdubbeling). Deze procedure is natuurlijk vrij bewerkelijk, en is dan ook veruit de langste van het programma. De procedure "StroreAchtNieuweBytes" zet de gevonden acht bytes in de buffer.



Hardcopy

Aangezien de hardcopy verdubbeld-gekanteld op de printer moet en printers nu eenmaal per horizontale rij van 8 hoog printen, zal het programma steeds een vertiacle rij van 4 bits breed uit het scherm moeten nemen. Dit zal 80 keer moeten gebeuren, want het scherm is 80x4=320 puntjes breed. Deze lus van 80 maal heet in de listing "BeginHorLus" (lussen worden voor de duidelijkheid ingesprongen weergegeven). U zult ook kunnen zien aan "cmp 80" dat deze inderdaad 80 maal wordt uitgevoerd. Wat er met zo'n rij van 4 breed (en 200 bits hoog) precies gebeurt, daar zorgt de procedure "FillBuffer" voor. Overigens, als Uw printer geen 8 puntjes hoge regels print, maar minder of misschien wel meer, dan zorgt de printerdriver ervoor dat dit gecamoufleerd wordt. Als programmamaker kunt U op dit punt dus gerust Uw verstand op nul zetten.

HaalVierbijVier

In deze procedures hebben de GEOS-variabelen a1L ($a2) en a1H ($a3) respectievelijk de functie van horizontale en vertikale teller binnen een blokje van vier bij vier. Per waarde van a1L wordt er naar de onder elkaar liggen bits in het blokje van vier bij vier gekeken. De vertikale teller a1H wordt dan dus gevarieerd (van 0 naar 3). Uit deze onder elkaar liggende punten wordt een byte gemaakt. Aangezien in een byte (bij HIRES) de bits NAAST elkaar ligen, is er hier sprake van kanteling. Maar hoe gaat dat nu precies? Uitgegaan wordt van een lege byte. Steeds als een puntje in de minibyte, zeg het N-de, aan is, dan moeten in de resultaatbyte 2 punten aangezet worden: de 2*N-de en

de (2*N +1)-de. Dit geeft een breedteverdubbelling. De lengteverdubbelling is veel eenvoudiger, twee opeenvolgende resultaatbytes zijn gewoon steeds hetzelfde.

TestPoint

Om te kijken of een puntje in het blokje van 4x4 aan is, wordt gebruik gemaakt van de GEOS-routine TestPoint. Deze routine ($c13f) heeft als input de X-coordinaat in register r3 ($0008/$0009) en de Y coordinaat in r11L ($0018). Als het puntje aan is, dan is de Carry 1, anders is de Carry 0. Als U de Listing goed leest, dan zult U zien dat r11L := 4*VertLus + a1H. (VertLus loopt van 0 tot 50 en geeft aan in welk blokje van een vertikale rij men zit). Aangezien het om blokjes van 4x4 gaat geeft 4*VertLus +a1H (de verticale teller binnen zo'n blokje) precies het totale aantal puntjes in verticale richting, ook wel bekend als Y- coördinaat. Deze berekening gebeurt vlak na het label "TweedeLus:". Zo ook was al eerder (ongeveer 10 regels onder "EersteLus:") berekend dat a3 = 4*HorLus +a1L. Horlus is daarin de variabele die aangeeft met welke rij van 4 breed bij 200 hoog men bezig is. U ruikt het al, a3 is de X-coordinaat. Later wordt dan a3 in r3 gezet. Dan zijn zowel r3 als r11L met de juiste waarden gevuld, zodat er naar TestPoint gesprongen kan worden.

GeoProgrammer

Zoals ik al zei: rekenen in machinetaal is niet bijster leuk. Maar ook als U het niet helemaal volgen kon, dan heeft u nog altijd de LISTING, en dat is ook niet niks. Hij bestaat uit drie delen (die altijd nodig zijn in de GeoProgrammer): de listing (hardprgLst) en de header (hardprgHdr) voor de GeoAssembler, en hardprg.ink voor de GeoLinker. De volhouders zonder GeoProgrammer zullen het deze keer erg moeilijk hebben. Ik raad hun dan ook van ganser harte aan om het programma maar bij een kennis die wel een GeoProgrammer heeft in te tikken. Niettemin, voor de onverschrokkennen heb ik de volgende aanwijzingen: maak gebruik van de ongewijzigde GEOS- PROGRAM MAKER. Het maakt niet uit waar er geassembleerd wordt, als dit maar in het gebied tussen $2000 en $5fff ligt. In de listing komen de symbolen "[" en "]" voor, deze staan respectievelijk voor LowByte en HighByte.

Peter Boncz

Listing 1.

```
;               hardprgLst
.include        geosSym
.include        geosMac
.ramsect
ApplicatieBuffer:
        .block  ≠640
PrDrvrBuffer:
        .block  ≠640
HorLus:
        .block  ≠1
VertLus:
        .block  ≠1
NieuwByte:
        .block  ≠8
.psect
InitRoutine:
        lda     dispBufferOn
        sta     a4L
        lda     ≠$80
        sta     dispBufferOn
        lda     ≠$00
        sei
        ldx     ≠$37
        stx     $01
        sta     $d020
        ldx     ≠$30
        stx     $01
        cli
        sta     $02
        LoadW   r6,PrntFilename
        jsr     GetFile
        cpx     ≠$00
        bne     EindeProgramma
        jsr     InitForPrint
        jsr     StartPrint
        cpx     ≠$00
        bne     EindeProgramma
        lda     ≠$ff
        sta     HorLus
        jsr     InitBuffer
        BeginHorLus:
                jsr     FillBuffer
                LoadW   r0,ApplicatieBuffer
                LoadW   r1,PrDrvrBuffer
                LoadW   r2,  ≠$01
                jsr     PrintBuffer
                inc     HorLus
                lda     HorLus
                cmp     ≠80
        bne             BeginHorLus
        jsr     StopPrint
EindeProgramma:
        lda     a4L
        sta     dispBufferOn
        jsr     $c22c
rts

InitBuffer:
        LoadW   a0,ApplicatieBuffer
        lda     ≠0
        ldx     ≠0
BeginLus:
        ldy     ≠0
        LaatsteLus:
                sta     (a0),y
                iny
                cpy      ≠128
                beq     MisschienStoppen
                cpy     ≠0
        bne             LaatsteLus
        inc     a0H
        inx
jmp             BeginLus
MisschienStoppen:
        cpx     ≠2
        bne     LaatsteLus
rts

FillBuffer:
        lda     HorLus
        cmp     ≠$ff
        beq     EindeRoutine
        lda     ≠50
        sta     VertLus
        BeginVertLus:
                jsr     HaalVierBijVier
                jsr     StoreAchtNieuweBytes
```

Listing 2.

```
;                               hardprgHdr
.include        geosSym                         ; Ook voor de header nodig.
.psect                                          ; Begin van de header.
.word           0                               ; Eerste 2 bytes altijd 0.
.byte           3                               ; Afmetingen Icon: 3 bytes
.byte           21                              ;  breed, 21 puntjes hoog.
.byte           (63+$80)                        ; 63 bytes icondata.
.byte           %11111111,%11111111,%11111111
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %10000000,%00000000,%00000001
.byte           %11111111,%11111111,%11111111
; dit was de icondata, 63 bytes. In Bitvorm weergegeven (voor
; de duidelijkheid. Nu is het icon helemaal leeg, op een kader
; na. Het Icon kan naar believen veranderd worden.
.byte           $83                             ; Geos File Type USR.
.byte           APPLICATION                     ; APPLICATION  = 5.
.byte           SEQUENTIAL                      ; SEQUENTIAL  = 0.
.word           InitRoutine                     ; laadadres.
.word           EndAdress                       ; eindadres.
.word           InitRoutine                     ; startadres.
.byte           "Hardcopy V1.2",0,0,0,0,0,0,0
.byte           "Peter Boncz",0,0,0,0,0,0,0,0,0
.byte           0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
.byte           0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
.byte           "Roep eerst 'hardcopy' aan"
.byte           "en druk dan op CTRL-H"
.byte           "voor een hardcopy",0,0,0,0,0,0,0
.byte           0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
.byte           0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Listing 1 (vervolg).

```
                dec     VertLus
                lda     VertLus
                cmp     #$ff
        bne             BeginVertLus
EindeRoutine:
        rts

TweedeMacht:
        tay
        lda     #$00
        sta     a2L
        sec
        MachtLus:
                rol     a2L
                dey
                cpy     #$ff
                clc
        bne     MachtLus
        lda     a2L
rts

HaalVierBijVier:
        lda     #$00
        sta     a1L
EersteLus:
        lda     a1L
        sta     a2H
        asl     a2H
        ldy     a2H
        lda     #$00
        sta     NieuwByte,y
        sta     NieuwByte+1,y
        sta     a1H
        sta     a3H
        lda     HorLus
        sta     a3L
        asl     a3L
        rol     a3H
        asl     a3L
        rol     a3H
        clc
        lda     a1L
        adc     a3L
        sta     a3L
        bcc     CarryClear
        inc     a3H
        clc
CarryClear:
        TweedeLus:
                ldx     #$00
                stx     r3H
                lda     VertLus
                sta     r11L
                asl     r11L
                asl     r11L
                lda     r11L
                clc
                adc     a1H
                sta     r11L
                lda     a3L
                sta     r3L
                lda     a3H
                sta     r3H
                lda     #$01
                jsr     TestPoint
                bcc     NietsTeDoen
                        jsr FillDoublePoint
```

```
        NietsTeDoen:
                inc     a1H
                lda     a1H
                cmp     #$04
        bne             TweedeLus
        inc     a1L
        lda     a1L
        cmp     #$04
bne             NaarEersteLus
rts
NaarEersteLus:
        jmp     EersteLus

StoreAchtNieuweBytes:
        lda     #$00
        sta     a0H
        clc
        lda     #50
        sbc     VertLus
        sta     a0L
        asl     a0L
        rol     a0H
        asl     a0L
        rol     a0H
        asl     a0L
        rol     a0H
        lda     #[(ApplicatieBuffer)
        clc
        adc     a0L
        sta     a0L
        bcc     geenoverflow
        inc     a0H
        clc
geenoverflow:
        lda     #](ApplicatieBuffer)
        adc     a0H
        sta     a0H
        ldy     #0
        WegschrijfLus:
                lda     NieuwByte,y
                sta     (a0),y
                iny
                cpy     #8
        bne             WegSchrijfLus
rts

FillDoublePoint:
        clc
        lda     a1H
        sta     a0L
        asl     a0L
        lda     a0L
        jsr     TweedeMacht
        sta     a0H
        inc     a0L
        lda     a0L
        jsr     TweedeMacht
        adc     a0H
        sta     a0H
        clc
        ldy     a2H
        lda     NieuwByte,y
        adc     a0H
        sta     NieuwByte,y
        sta     NieuwByte+1,y
rts
EndAdress:
```

listing 3.

```
                        hardprg.ink
;
.output         hardcopy.prg        ; naam voor de outputfile.
.header         hardprgHdr.rel      ; naam van de file met
                                    ;  de header.
.seq                                ; dit is een sequentiele
                                    ;  applicatie.
.psect          $6000               ; begin programmacode: $6000
.ramsect        $6300               ; begin variabelen: $6300
hardprgLst.rel                      ; naam van de file met de zg
                                    ; 'relocatable' machinekode
```

PRINT OUT C-16 met o.a.Dropper

Checksum c16

```
10    rem ******************************
20    rem syntax.checksum
30    rem voor c-16 & plus/4
40    rem
50    rem syntax testen met 'sys 1536'
60    rem
70    rem v.851128.16        jan bodzinga
80    rem ******************************
90    i=1536        :rem beginadres
100   reada:ifa>=0then pokei,a:i=i+1:got
      o100
110   print"data[SPACE]is[SPACE]weggezet"
120   print"cheksum[SPACE]printen[SPACE]
      met[SPACE]'sys[SPACE]1536'
130   end
200   data 165, 43,166, 44,133
210   data  31,134, 32,169,147
220   data  32,210,255,160,  0
230   data 240,  3, 32, 73,  6
240   data  32, 73,  6,208,  1
250   data  96, 72,152, 32,131
260   data   6,168,104,234, 32
270   data  81,  6, 32, 73,  6
280   data 240, 12,201, 32,240
290   data 247, 24,101,252,133
300   data 252, 76, 37,  6,166
310   data 252,169,  0,132,253
320   data  32, 95,164,169, 13
330   data  32,210,255,164,253
340   data  76, 17,  6,200,208
350   data   2,230, 32,177, 31
360   data  96,162,  0,189,123
370   data   6,240,  6, 32,210
380   data 255,232,208,245, 32
390   data  73,  6,170, 32, 73
400   data   6,132,253, 32, 95
410   data 164,162,  3,169, 32
420   data  32,210,255,202,208
430   data 250,169,  0,133,252
440   data 164,253, 96, 82, 69
450   data  71, 69, 76, 32,  0
460   data   0, 72,138, 72, 32
470   data 225,255,240,251,104
480   data 170,104, 96, -1
```

** EINDE LISTING checks16

Checksum Checksum C-16

REGEL 10	249	REGEL 300	118
REGEL 20	247	REGEL 310	204
REGEL 30	121	REGEL 320	165
REGEL 40	143	REGEL 330	252
REGEL 50	75	REGEL 340	106
REGEL 60	143	REGEL 350	98
REGEL 70	8	REGEL 360	163
REGEL 80	249	REGEL 370	45
REGEL 90	103	REGEL 380	0
REGEL 100	2	REGEL 390	58
REGEL 110	245	REGEL 400	108
REGEL 120	237	REGEL 410	159
REGEL 130	128	REGEL 420	245
REGEL 200	210	REGEL 430	202
REGEL 210	208	REGEL 440	176
REGEL 220	142	REGEL 450	12
REGEL 230	1	REGEL 460	54
REGEL 240	3	REGEL 470	43
REGEL 250	157	REGEL 480	1
REGEL 260	155		
REGEL 270	215		
REGEL 280	186		
REGEL 290	248		

Programma omzet16

Dit programma geschreven door R. Luttjeboer uit Winschoten maakt het hires beeldscherm geschikt voor uitgave naar de printer. Omdat het hires beeldscherm in kolommetjes van 8x8 bits is opgebouwd is deze omzetting nodig. Het programma dat in machinetaal is geschreven spiegelt elk kolommetje om de diagonaal die van linksboven naar rechtsonder door dit kolommetje loopt. Door een 'sys 1536' kommando wordt het beeldscherm omgezet. Herstellen gebeurt met hetzelfde kommando. Het programma is bedoeld als onderdeel van een ander programma.

```
10    rem  printer omzetter  voor c-16
20    rem  door r. luttjeboer
30    rem  uit winschoten
40    rem  tel: 05970-22161
50    rem
60    rem  contact gezocht met andere c-
      16
70    rem  gebruikers voor uitwisseling
      van
80    rem  programma's en ervaringen
90    rem
100   b=1536
110   reada:ifa=-1thengoto200:elsepokeb,
      a:b=b+1:goto110
120   data169,0,141,86,3,141,20,6,141,58
      ,6,169,32,141,21,6
130   data141,59,6,173,64,63,141,78,3,23
      8,20,6,238,23,6,173
140   data20,6,201,0,208,3,238,21,6,173,
      23,6,201,86,208
150   data227,169,78,141,23,6,24,46,78,3
      ,46,64,63,238,55,6
160   data173,55,6,201,86,208,239,169,78
      ,141,55,6,238,58,6
170   data173,58,6,201,0,208,3,238,59,6,
      238,86,3,173,86,3
180   data201,8,208,211,169,0,141,86,3,1
      73,21,6,201,63,208
190   data165,173,20,6,201,64,208,158,96,-1
200   scnclr:print"[SPACE]voor[SPACE]omz
      etting[SPACE]grafisch[SPACE]beelds
      cherm"
210   print"[SPACE]'graphic1,0:sys1536'[
      SPACE]commando[SPACE]geven."
220   print"[SPACE]voor[SPACE]herstellen
      [SPACE]van[SPACE]beeldscherm"
240   print"[SPACE]nogmaals[SPACE]'sys[S
      PACE]1536'[SPACE]commando[SPACE]ge
      ven."
```

** EINDE LISTING omzet16

Checksum printer omzetter

REGEL 10	74	REGEL 130	255
REGEL 20	67	REGEL 140	26
REGEL 30	131	REGEL 150	222
REGEL 40	220	REGEL 160	228
REGEL 50	143	REGEL 170	160
REGEL 60	27	REGEL 180	181
REGEL 70	84	REGEL 190	224
REGEL 80	51	REGEL 200	107
REGEL 90	143	REGEL 210	175
REGEL 100	195	REGEL 220	28
REGEL 110	19	REGEL 240	60
REGEL 120	185		

Programma hardcopy c16

Ook dit programma werd geschreven door R. Lutjeboer uit Winschoten en dit programma is een kompleet hires hardcopyprogramma. Het maakt gebruik van de printer omzetter. De gebruikte instelcodes zijn voor de General Electric TXP-1000 maar kunnen vervangen worden door commando's voor andere printers met grafische mogelijkheden. Het programma kan na bijvoorbeeld een tekenprogramma gebruikt worden mits het tekenprogramma na afloop het grafische beeld niet uitwist. Ook kan een subroutine met tekeninstructies ingebouwd worden.

```
10    rem   printer omzetter  voor c-16
20    rem   door r. luttjeboer
30    rem   uit winschoten
40    rem   tel: 05970-22161
50    rem
60    rem   contact gezocht met andere c-16
70    rem   gebruikers voor uitwisseling
      van
80    rem   programma's en ervaringen
90    rem
100   b=1536
110   reada:ifa=-1thengoto200:elsepokeb,
      a:b=b+1:goto110
120   data169,0,141,86,3,141,20,6,141,58
      ,6,169,32,141,21,6
130   data141,59,6,173,64,63,141,78,3,23
      8,20,6,238,23,6,173
140   data20,6,201,0,208,3,238,21,6,173,
      23,6,201,86,208
150   data227,169,78,141,23,6,24,46,78,3
      ,46,64,63,238,55,6
160   data173,55,6,201,86,208,239,169,78
      ,141,55,6,238,58,6
170   data173,58,6,201,0,208,3,238,59,6,
      238,86,3,173,86,3
180   data201,8,208,211,169,0,141,86,3,1
      73,21,6,201,63,208
190   data165,173,20,6,201,64,208,158,96,-1
200   scnclr:print"[SPACE]voor[SPACE]omz
      etting[SPACE]grafisch[SPACE]beelds
      cherm"
210   print"[SPACE]'graphic1,0:sys1536'[
      SPACE]commando[SPACE]geven."
220   print"[SPACE]voor[SPACE]herstellen
      [SPACE]van[SPACE]beeldscherm"
240   print"[SPACE]nogmaals[SPACE]'sys[S
      PACE]1536'[SPACE]commando[SPACE]ge
      ven."
```

** EINDE LISTING hardco16

Checksum Hardcopy

REGEL 10	74	REGEL 130	255
REGEL 20	67	REGEL 140	26
REGEL 30	131	REGEL 150	222
REGEL 40	220	REGEL 160	228
REGEL 50	143	REGEL 170	160
REGEL 60	27	REGEL 180	181
REGEL 70	84	REGEL 190	224
REGEL 80	51	REGEL 200	107
REGEL 90	143	REGEL 210	175
REGEL 100	195	REGEL 220	28
REGEL 110	19	REGEL 240	60
REGEL 120	185		

Programma dropper c16

Peter Boersma is ook de maker van dit spel Peter stuurt regelmatig programma's naar ons op en al diverse malen zijn programma's van hem geplaatst de eerste is al weer van twee jaar geleden.

Dropper is een spel waarbij de speler met joystick 1 moet proberen een balletje tussen stijgende balken door te manoeuvreren. Daarbij zijn 3 fasen te onderscheiden. 1. de begin fase. veel balken; zorg dat je nier verpletterd wordt tegen het plafond. 2. tussen fase; het aantal balken wordt minder; gewoon zo lang mogelijk volhouden. 3 de eindfase; er zijn nauwelijks genoeg balken; blijf er zolang mogelijk op liggen zodat je kunt zien waar de volgende komt.

```
1     rem -c-16————————-c-
      16-
2     rem ——-    d r o p p e r   ——
      —
3     rem -c-16————————-c-
      16-
4     rem - door:  peter boersma (mei'88
      ) -
5     rem -alias:social software service
      s -
6     rem - voor:  commodore info
      ' -
7     rem - (sss 1988)
      -
8     rem - nr. 12-107  enschede
      -
9     rem -c-16————————-c-
      16-
10    scnclr:color0,1:color4,1:color1,8,
      4
20    printchr$(27)"n";chr$(142)chr$(8);
30    print"score:";sc;tab(11)"[CTRL-9]d
      ropper[CTRL-0][SPACE](c)sss'88[SPA
      CE]high";hi
40    printchr$(27)"t":gosub 380
50    color0,1:color4,1:color1,8,4
60    for a=1 to 24:l$=""
70    if rnd(0)>.4 then goto 100
80    w=int(rnd(0)*30):l=int(rnd(0)*10)+
      1
90    l$="[CTRL-9]":for b=1 to l:l$=l$+"
      [SPACE]":nextb
100   printtab(w)l$:nexta:p=3492
110   geta$:rem poke p,32
120   if a$="6" and p<4071 then p=p+1
130   if a$="d" and p>3112 then p=p-1
140   if p<4031 and peek(p+00)=32 then p
      =p+40
```

```
150  if peek(p+00)=160and p>3152 then p=p-40
160  if p<3152 or p>3992 then goto 270
170  poke p,81:sc=sc+1:gosub 240
180  if rnd(0)>1-sc/1000then goto 210
190  w=int(rnd(0)*30):l=int(rnd(0)*10)+1
200  l$="[CTRL-9]":for b=1 to l:l$=l$+"[SPACE]":nextb
210  k=k+1:if k=7 then k=3
220  color1,8,k:printtab(w)l$:l$="":goto 110
230  rem —— score ————————-
240  s$=str$(sc):for a=1 to len(s$)
250  poke 3077+a,asc(mid$(s$,a,1)):next:return
260  rem —— einde ————————-
270  scnclr:h=0:if sc>hi then hi=sc:h=1
280  h$=str$(hi):fora=1 to len(h$)
290  poke 3104+a,asc(mid$(h$,a,1)):next
300  c$="":if h=1 then c$="nieuwe[SPACE]high-score[SPACE]!"
310  char 1,10,11,c$:print
320  printtab(10)"[2xCRSR-DOWN][SPACE]score:";sc
330  printtab(10)"[2xCRSR-DOWN][SPACE]high[SPACE]:";hi
340  print"[3xCRSR-DOWN][10xSPACE]druk[SPACE]op[SPACE]'return'"
350  geta$:ifa$<>chr$(13)then350
360  sc=0:goto 10
370  rem —— introductie ——————-
380  scnclr:color0,1:color4,3,3:color1,2
390  color1,5,7:print"[SPACE][COM-A]CCCI[COM-A]CCCIUCCCI[COM-A]CCCI[COM-A]CCCIUCCCI[COM-A]CCCI(c)[2xSPACE]"
400  color1,5,5:print"[SPACE]B[3xSPACE]BB[3xSPACE]BB[3xSPACE]BB[3xSPACE]BB[3xSPACE]BB[3xSPACE]BB[3xSPACE]Bsss[2xSPACE]"
410  color1,5,4:print"[SPACE]B[3xSPACE]B[COM-Q]C[COM-R]CKB[3xSPACE]B[COM-Q]CCCK[COM-Q]CCCK[COM-Q]CCCK[COM-Q]C[COM-R]CK'88[2xSPACE]"
420  color1,5,3:print"[SPACE]B[3xSPACE]BB[SPACE]B[2xSPACE]B[3xSPACE]BB[4xSPACE]B[4xSPACE]B[4xSPACE]B[SPACE]B[7xSPACE]"
430  color1,5,1:print"[SPACE][COM-Z]CCCK[COM-E][SPACE]JC[SPACE]JCCCK[COM-E][4xSPACE][COM-E][4xSPACE]JCCCK[COM E][SPACE]JC[6xSPACE]"
440  color1,5,2:printtab(17)"[3xCRSR-DOWN][CTRL-9]dropper"
450  color1,5,3:printtab(15)"[CRSR-DOWN][CTRL-9]is[SPACE]een[SPACE]spel"
460  color1,5,4:printtab(13)"[CRSR-DOWN][CTRL-9]waarbij[SPACE]jij[SPACE]een"
470  color1,5,5:printtab(11)"[CRSR-DOWN][CTRL-9]balletje[SPACE]door[SPACE]balken"
480  color1,5,6:printtab(08)"[CRSR-DOWN][CTRL-9]moet[SPACE]sturen[SPACE]zonder[SPACE]dat[SPACE]het"
490  color1,5,7:printtab(06)"[CRSR-DOWN][CTRL-9]balleje[SPACE]de[SPACE]grond[SPACE]of[SPACE]het[SPACE]plafond"
500  color1,5,6:printtab(03)"[CRSR-DOWN][CTRL-9]raakt.[SPACE]besturing[SPACE]gaat[SPACE]via[SPACE]joystick[SPACE]1"
510  color1,5,5:printtab(01)"[CRSR-DOWN][CTRL-9]druk[SPACE]op[SPACE]de[SPACE]spatiebalk[SPACE]om[SPACE]te[SPACE]beginnen...":q=0
520  for a=1 to 100:next a
530  if q>4031 then poke q,32:q=0
540  get a$:if a$=chr$(32) then return
550  if q=0then q=3332
560  if peek(q+40)=160or peek(q+40)=32 then poke q,32:q=q+40:poke q,81:goto 520
570  d=int(rnd(0)*3-1):if d=0then 570
580  poke q,32:q=q+d:poke q,81:if peek(q+40)=32 or peek(q+40)=160then goto 560
590  for a=1 to 100:next a:goto 580
600  rem -sss——————————-'88-
610  rem ——-  d r o p p e r  ——-
620  rem -c-16——————————-c-16-
```

** EINDE LISTING dropper16

Checksum dropper

REGEL 1	162	REGEL 110	223	REGEL 300	8	REGEL 490	180
REGEL 2	33	REGEL 120	110	REGEL 310	146	REGEL 500	54
REGEL 3	162	REGEL 130	118	REGEL 320	179	REGEL 510	241
REGEL 4	163	REGEL 140	144	REGEL 330	82	REGEL 520	215
REGEL 5	23	REGEL 150	196	REGEL 340	19	REGEL 530	226
REGEL 6	58	REGEL 160	170	REGEL 350	79	REGEL 540	148
REGEL 7	14	REGEL 170	204	REGEL 360	156	REGEL 550	51
REGEL 8	129	REGEL 180	27	REGEL 370	3	REGEL 560	76
REGEL 9	162	REGEL 190	188	REGEL 380	89	REGEL 570	33
REGEL 10	94	REGEL 200	0	REGEL 390	31	REGEL 580	48
REGEL 20	33	REGEL 210	243	REGEL 400	134	REGEL 590	55
REGEL 30	225	REGEL 220	78	REGEL 410	40	REGEL 600	132
REGEL 40	20	REGEL 230	67	REGEL 420	69	REGEL 610	33
REGEL 50	60	REGEL 240	226	REGEL 430	72	REGEL 620	72
REGEL 60	79	REGEL 250	118	REGEL 440	131		
REGEL 70	155	REGEL 260	44	REGEL 450	236		
REGEL 80	188	REGEL 270	206	REGEL 460	248		
REGEL 90	0	REGEL 280	199	REGEL 470	102		
REGEL 100	55	REGEL 290	154	REGEL 480	235		

Checksum C-64

Syntax Checksum

Het overtikken van een listing kan een heel karwei zijn en als u een beetje normaal mens bent dan maakt u daarin beslist een aantal fouten. Nu is niets moeilijker om de fouten uit je eigen werk te halen. Al geruime tijd heeft Jan Bodzinga hiervoor een zgn. Checksum-programma geschreven. Om de vele nieuwe lezers van Commodore-info te helpen volgt hieronder nog een keer een volledige uitleg over de werking van dit programma, waarmee het, hoe vreemd dat misschien ook lijkt, echt mogelijk is om met behulp van dit programma de fouten in elke door ons geplaatste listing op te sporen.

Hiervoor gaat u als volgt te werk:

1. U tikt de listing heel zorgvuldig over en SAVEt hem voordat u het programma RUNt op een diskette of cassette.
2. U tikt het RUN commando in. Mocht het programma de boodschap 'FOUT in dataregels!' geven dan heeft u een fout bij het overtikken gemaakt. Herstel dan de fout en SAVE de verbeterde versie. Mocht het programma met de boodschap 'data is weggezet checksum testen met sys...'komen dan is tot dusver alles goed. Het programma is nu in een stukje machinetaalgeheugen gezet. Als u het NEW commando geeft blijft het toch in de computer staan.

Alle door ons geplaatste programma's zijn in Basic geschreven.

Als u een programma heeft overgetikt SAVE het eerst, mocht er iets mis gaan dan hoeft u niet de gehele listing opnieuw te gaan intikken. Als u nu een programma op fouten wilt gaan controleren dan kunt u dat in het geheugen laden(wel eerst het checksum programma hebben gerund). Vervolgens typt u zonder het programma te runnen de opdracht sys 49152(c-64) of sys 1536 (c-16 en plus/4)in.

Als alles goed is gegaan loopt er nu een rij regelnummers over het scherm met getallen erachter. Dezelfde lijst staat ook achter elk door ons geplaatste programma. Wijkt nu een nummer achter een regelnummer af van het nummer dat in het blad staat dan heeft u in die regel iets anders ingetikt dan er in het blad stond. U kunt de stroom getallen d.m.v. de RUN/STOP toets pauzeren en weer vervolgen met de F1 of F7 toets. Het is uitermate belangrijk dat u goed met dit programma overweg kunt en mocht u het niet goed werkend krijgen bel dan gerust even met onze listingservice telefoonlijn. (Maandag 17.00 - 21.00 uur. Telefoonnummer 02155-25162.)

```
1    rem ******************************
2    rem basic loader "SYNTAX.CHECKSUM"
3    rem na de commando's "run" en "new"
4    rem blijft dit programma in het ge-
5    rem heugen. laad het te testen pro-
6    rem gramma en tik daarna sys 49152.
7    rem ******************************
10   i=49152   :rem beginadres
20   reada:ifa<0then40:rem data ingelezen
30   pokei,a:i=i+1:b=b+a:goto20
40   if b<>16844thenprint"[SHIFT-CLR]fo
     ut[SPACE]in[SPACE]dataregels!":b=0
     :end
50   poke49184,148:poke49185,192
55   i=49300
60   read a: ifa<0then80
70   pokei,a:b=a+b:i=i+1:goto60
80   if b<>20068thenprint"[SHIFT-CLR]fo
     ut[SPACE]in[SPACE]dataregels![SPAC
     E](vanaf[SPACE]regel[SPACE]240)":b
     =0:end
90   print"data[SPACE]is[SPACE]weggezet"
95   print"checksum[SPACE]testen[SPACE]
     met[SPACE]sys49152"
100  data 165,43,166,44,133,163,134,164
     ,169,147
110  data 32,210,255,160,0,240,3,32,73,192
120  data 32,73,192,208,1,96,32,225,255,208
130  data 3,76,116,164,32,81,192,32,73,192
140  data 240,12,201,32,240,247,24,101,
     167,133
150  data 167,76,37,192,166,167,169,0,1
     32,168
160  data 32,205,189,169,13,32,210,255,
     164,168
170  data 76,17,192,200,208,2,230,164,1
     77,163
180  data 96,162,0,189,123,192,240,6,32,210
190  data 255,232,208,245,32,73,192,170
     ,32,73
200  data 192,132,168,32,205,189,162,3,
     169,32
210  data 32,210,255,202,208,250,169,0,
     133,167
220  data 164,168,96,82,69,71,69,76,32,0
230  data  -1
240  data 165,197,201,3,240,7,201,4,240
250  data 6,76,148,192,76,34,192,169
260  data 147,32,210,255,76,161,192
270  data -1
```

** EINDE LISTING checksum 64 **

Chechsum Checksum C-64

REGEL 1	249	REGEL 150	96
REGEL 2	84	REGEL 160	127
REGEL 3	105	REGEL 170	71
REGEL 4	2	REGEL 180	223
REGEL 5	246	REGEL 190	73
REGEL 6	152	REGEL 200	79
REGEL 7	249	REGEL 210	109
REGEL 10	157	REGEL 220	106
REGEL 20	64	REGEL 230	225
REGEL 30	38	REGEL 240	16
REGEL 40	57	REGEL 250	163
REGEL 50	14	REGEL 260	92
REGEL 55	251	REGEL 270	22
REGEL 60	192		
REGEL 70	42		
REGEL 80	244		
REGEL 90	245		
REGEL 95	237		
REGEL 100	183		
REGEL 110	158		
REGEL 120	232		
REGEL 130	183		
REGEL 140	96		

PRINT OUT C-64 met o.a. Key maze

Programma golf 64

Golf een heerlijke zomerse sport, zelfs op de computer is dit spel al in verschillende variaties aanwezig. Toch heeft M. Boorsma een golf programma geschreven, dat nog niet eerder is verschenen, een duidelijke uitleg welke toetsen te gebruiken is in het programma opgenomen.

```
10    rem *******************************
20    rem * gemaakt door :              *
30    rem *                             *
40    rem * michiel boorsma             *
50    rem *                             *
60    rem * voor :                      *
70    rem *                             *
80    rem * commodore info              *
90    rem *                             *
95    rem *programma starten met run97 *
96    rem *******************************
97    hs=10[KWADRAAT PIJL]5
100   gosub1000:poke53280,12:poke53281,5
      :poke649,1:poke650,132
110   s=2:sc=1024:co=55296:xc=0:a=1:cl=1
      :ho=1:ac=0
120   un=32:uc=5:nu=32:cu=5:yb=23:xb=int
      (rnd(1)*38+1):xc=int(rnd(1)*3+2)
130   hi=0:print"[SHIFT-CLR][CTRL-6]";:f
      ori=1to12+4*ho:xc=int(rnd(1)*40)
135 yc=int(rnd(1)*22+3):pokesc+40*yc+x
      c,88:pokeco+yc*40+xc,0:next
140   xc=int(rnd(1)*5+3)
150   fori=1toxc:print"[CRSR-DOWN]";:nex
      t:fori=2toint(rnd(1)*20+2)
160   print"[CRSR-RIGHT]";:next:print"[C
      TRL 8][COM-P][COM-O][COM-I][CTRL-9
      ][COM-T][COM-Y][COM-T][CTRL-0][COM
       I][COM-P][COM-`][CTRL-4][COM-G][C
      RSR-LEFT][CRSR-UP][SHIFT-|][2xCRSR
      -LEFT][CRSR-UP][CTRL-9]"ho"[CTRL-0
      ][CRSR-LEFT][SHIFT-|][3xCRSR-LEFT]
      [SPACE][2xCRSR-DOWN][2xCRSR-RIGHT]
      [CTRL-8][COM-P][COM-O][COM-`]";
170   print"[CRSR-DOWN][15xCRSR-LEFT][CO
      M T][CTRL-9][9xSPACE]Q[4xSPACE][CO
      M T][CRSR-DOWN][13xCRSR-LEFT][COM-
      V][2xSPACE][COM-D][COM-F][3xSPACE]
      [COM-D][COM-F]";
180   print"[CRSR-DOWN][12xCRSR-LEFT][CO
      M T][COM-Y][COM-T][CTRL-9][COM-O][
      COM-P][CRSR-RIGHT][COM-J][5xSPACE]
      [CRSR-DOWN][5xCRSR-LEFT][4xSPACE][
      CTRL-0][SPACE]";
190   print"[CRSR-DOWN][6xCRSR-LEFT][COM
       T][COM-Y][COM-T][CTRL-9][COM-O][C
      TRL 0][2xSPACE]"
200   print"[HOME]":fori=1to15+int(rnd(1
      )*6):print"[CRSR-DOWN]";:next
210   fori=1toint(rnd(1)*35):print"[CRSR
      -RIGHT]";:next:print"[CTRL-7][4xCO
      M +][2xCOM-|][CRSR-DOWN][4xCRSR-LE
      FT][2xCOM-+][COM—][CRSR-DOWN][3xC
      RSR-LEFT][COM-+]"
220   print"[HOME]":fori=1to15+int(rnd(1
      )*7):print"[CRSR-DOWN]";:next
225   fori=1toint(rnd(1)*35):print"[CRSR
      -RIGHT]";:next:print"[2xCOM-+][CRS
      R-DOWN][2xCRSR-LEFT][3xCOM-+]"
230   print"[HOME]":fori=1to15+int(rnd(1
      )*6):print"[CRSR-DOWN]";:next
235   forx=1toint(rnd(1)*30):print"[CRSR
      -RIGHT]";:next:print"[2xCOM-|][3xC
      OM +][COM-|][CRSR-DOWN][5xCRSR-LEF
      T][4xCOM-+][CRSR-DOWN][5xCRSR-LEFT
      ][COM-|][3xCOM-+][3xCOM-|]"
240   yc=25:xc=20:hh=1:un=32:uc=1
250   poke197,0:pokeco+yb*40+xb,1:pokesc
      +yb*40+xb,81
260   print"[HOME][CTRL-1][2xSPACE]club[
      SPACE]?[17xSPACE][HOME]";
270   geta$:ifa$<"1"ora$>"9"then270
280   cl=val(a$):printtab(6)cl
290   print"[HOME]"tab(10)"[CTRL-4]krach
      t[SPACE]:";:ifhh=1thenprint"zacht[
      2xSPACE]"
300   ifhh=2thenprint"normaal"
310   ifhh=3thenprint"hard[3xSPACE]"
320   c=93:ifa=2thenc=77
330   ifa=3thenc=64
340   ifa=4thenc=78
350   geta$
355   ifa$<>"`"anda$<>":"anda$<>";"anda$
      <>"/"then440
360   pokeco+yc*40+xc,uc
370   pokesc+yc*40+xc,un:xc=xc+(a$=":")-
      (a$=";"):yc=yc+(a$="`")-(a$="/")
380   ifyc<1thenyc=1
390   ifyc>24thenyc=24
400   ifxc<0thenxc=0
410   ifxc>39thenxc=39
420   un=peek(sc+yc*40+xc):uc=peek(co+yc
      *40+xc)
430   ifun=81thenyc=yb+1:goto420
440   pokeco+yc*40+xc,2:pokesc+yc*40+xc,
      c
450   a=a+(a$=".")-(a$=","):ifa<1thena=4
460   ifa>4thena=1
470   ifa$="h"thenhh=3
480   ifa$="n"thenhh=2
490   ifa$="z"thenhh=1
500   ifa$>="1"anda$<="9"thencl=val(a$):
      print"[CTRL-1][HOME]"tab(6)cl
510   ifa$<>"[SPACE]"then290
520   dx=0:dy=0:ifa=1andyc<>ybthen660
530   ifa=1andabs(xc-xb)>1then660
540   ifa=1andxc<xbthendx=1:goto690
550   ifa=1andxc>xbthendx=-1:goto690
560   ifa=2and(abs(xc-xb)<>1orabs(yc-yb)
      <>1)then660
570   ifa=2andxc<xbandyc>ybthendx=1:dy=-
      1:goto690
580   ifa=2andxc>xbandyc<ybthendx=-1:dy=
      1:goto690
590   ifa=3andxc<>xbthen660
600   ifa=3andabs(yb-yc)>1then660
610   ifa=3andyc<ybthendy=1:goto690
620   ifa=3andyc>ybthendy=-1:goto690
630   ifa=4and(abs(xc-xb)<>1orabs(yc-yb)
      <>1)then660
640   ifa=4andxc>xbandyc>ybthendx=-1:dy=
      -1:goto690
650   ifa=4andxc<xbandyc<ybthendx=1:dy=1
      :goto690
660   print"[HOME][CTRL-2][SPACE]mis[SPA
      CE]![19xSPACE]":fori=1to1000:next:
      print"[HOME][7xSPACE]"
```

```
670     pokesc+yc*40+xc,un:pokeco+yc*40+xc
        ,uc
680     goto260
690     d=int((rnd(1)*(cl/3))*3+1)*hh:ifnu
        =102thend=int(d/2)
700     hi=hi+1
710     pokeco+yb*40+xb,cu:pokesc+yb*40+xb
        ,nu:yb=yb+dy:xb=xb+dx
720     ifxb<1orxb>39oryb<2oryb>24then740
730     goto760
740     print"[HOME][CTRL-2][2xSPACE]bal[S
        PACE]uit[17xSPACE]":fori=1to1500:n
        ext:nu=32
750     yb=23:xb=int(rnd(1)*37+1):goto250
760     cu=peek(co+yb*40+xb):nu=peek(sc+yb
        *40+xb)
770     ifnu=88thend=1:xb=xb-dx:yb=yb-dy:g
        oto720
780     pokeco+yb*40+xb,1:pokesc+yb*40+xb,
        81
790     fori=0to20:next
800     ifnu=209andd<3then850
810     if(nu=102ornu=104)and(d<4orhh<3)th
        end=1
820     d=d-1:ifd>0then710
830     pokesc+yc*40+xc,un:pokeco+yc*40+xc
        ,uc
840     goto260
850     print"[CTRL-1][HOME]hole[SPACE]in"
        hi"[CRSR-LEFT][SPACE]keer[21xSPACE
        ]"
860     ac=ac+hi
870     fori=1to2000:next:print"[SHIFT-CLR
        ][3xCRSR-DOWN][3xCRSR-RIGHT]"ho"[C
        RSR-LEFT]e[SPACE]hole[SPACE]gehaal
        d":print"[COM-4][CRSR-DOWN][4xCRSR
        -RIGHT]score[SPACE]tot[SPACE]nu[SP
        ACE]toe"ac
880     ho=ho+1:ifho>9then1200
890     fori=1to5000:next:goto120
1000    poke53280,6:poke53281,6
1005    print"[SHIFT-CLR]"tab(10)"[CTRL-8]
        ***[SPACE]golf[SPACE]***":print"[C
        RSR-RIGHT][CRSR-DOWN][CTRL-1]dit[S
        PACE]golfterrein[SPACE]heeft[SPACE
        ]negen[SPACE]holen."
1010    print"[CRSR-RIGHT]wanneer[SPACE]cl
        ub[SPACE]wordt[SPACE]gevraagd[SPAC
        E]geef[SPACE]dan[4xSPACE]een[SPACE
        ]nummer[SPACE]in[SPACE]van[SPACE]1
        [SPACE]tot[SPACE]9."
1020    print"[CRSR-DOWN][COM-5][SPACE][CO
        M `][CRSR-LEFT][CRSR-DOWN][CTRL-9]
        `[CTRL-0][SPACE]=[SPACE]club[SPACE
        ]naar[SPACE]boven
1030    print"[CTRL-5][CRSR-RIGHT][CTRL-9]
        [[CTRL-0][SPACE]=[SPACE]club[SPACE
        ]naar[SPACE]links
1040    print"[CTRL-5][CRSR-RIGHT][CTRL-9]
        ][CTRL-0][SPACE]=[SPACE]club[SPACE
        ]naar[SPACE]rechts
1050    print"[CTRL-5][CRSR-RIGHT][CTRL-9]
        /[CTRL-0][SPACE]=[SPACE]club[SPACE
        ]naar[SPACE]beneden
1060    print"[COM-6][SPACE][COM-`][4xSPAC
        E][COM-`][6xCRSR-LEFT][CRSR-DOWN][
        CTRL-9]<[CTRL-0][SPACE]en[SPACE][C
        TRL 9]>[CTRL-0][SPACE]draaien[SPAC
        E]club
1070    print"[COM-5][SPACE][COM-`][CRSR-L
        EFT][CRSR-DOWN][CTRL-9]z[CTRL-0][S
        PACE]voor[SPACE]een[SPACE]zacht[SP
        ACE]schot
1080    print"[CRSR-RIGHT][CTRL-9]n[CTRL-0
        ][SPACE]voor[SPACE]een[SPACE]norma
        al[SPACE]schot
1090    print"[CRSR-RIGHT][CTRL-9]h[CTRL-0
        ][SPACE]voor[SPACE]een[SPACE]hard[
        SPACE]schot
1100    print"[CTRL-8][SPACE][6xCOM-`][7xC
        RSR-LEFT][CRSR-DOWN][SPACE][CTRL-9
        ]spatie[CTRL-0][SPACE]=[SPACE]sla[
        SPACE]bal
1110    print"[CTRL-6][3xCRSR-DOWN][CRSR-R
        IGHT][22xCOM-P]"
1120    print"[CRSR-RIGHT][CTRL-9]druk[SPA
        CE]toets[SPACE]voor[SPACE]start."
1130    print"[CRSR-RIGHT][CTRL-0][22xCOM-
        Y][HOME]"
1140    geta$:ifa$=""then1140
1150    return
1200    print"[SHIFT-CLR][5xCRSR-DOWN][4xC
        RSR-RIGHT][CTRL-1]je[SPACE]hebt[SP
        ACE]de[SPACE]hele[SPACE]baan[SPACE
        ]afgelegd"
1210    print"[CTRL-3][CRSR-DOWN][4xCRSR-R
        IGHT]score"ac
1220    ifac>=hsthenprint"[CTRL-2][CRSR-DO
        WN][4xCRSR-RIGHT]beste[SPACE]score
        "hs:goto1260
1230    hs=ac
1240    print"[CTRL-2][CRSR-DOWN][4xCRSR-R
        IGHT]het[SPACE]is[SPACE]de[SPACE]b
        este[SPACE]score[SPACE]die"
1250    print"[4xCRSR-RIGHT]tot[SPACE]nu[S
        PACE]toe[SPACE]gehaald[SPACE]is"
1260    print"[2xCRSR-DOWN][4xCRSR-RIGHT][
        COM-4]nog[SPACE]een[SPACE]keer[SPA
        CE]?"
1270    geta$:ifa$=""then1270
1280    ifa$<>"n"then100
1290    print"[SHIFT-CLR][COM-5]":poke5328
        1,6:end
```

EINDE LISTING golf 64

Checksum golf

REGEL 10	123
REGEL 20	75
REGEL 30	227
REGEL 40	241
REGEL 50	227
REGEL 60	99
REGEL 70	227
REGEL 80	180
REGEL 90	227
REGEL 95	245
REGEL 96	123
REGEL 97	145
REGEL 100	228
REGEL 110	30
REGEL 120	76
REGEL 130	95
REGEL 135	50
REGEL 140	78
REGEL 150	196
REGEL 160	9
REGEL 170	27
REGEL 180	130
REGEL 190	246
REGEL 200	207
REGEL 210	44
REGEL 220	208
REGEL 225	115
REGEL 230	207
REGEL 235	76
REGEL 240	244
REGEL 250	130
REGEL 260	51
REGEL 270	219
REGEL 280	32
REGEL 290	104
REGEL 300	141
REGEL 310	163
REGEL 320	85
REGEL 330	183
REGEL 340	189
REGEL 350	6
REGEL 355	239
REGEL 360	136
REGEL 370	27
REGEL 380	49
REGEL 390	153
REGEL 400	45
REGEL 410	163
REGEL 420	93
REGEL 430	13
REGEL 440	147
REGEL 450	243
REGEL 460	124
REGEL 470	74
REGEL 480	79
REGEL 490	90
REGEL 500	108
REGEL 510	218
REGEL 520	173
REGEL 530	106
REGEL 540	206
REGEL 550	119
REGEL 560	157
REGEL 570	203
REGEL 580	203
REGEL 590	60
REGEL 600	110
REGEL 610	211
REGEL 620	124
REGEL 630	159
REGEL 640	118
REGEL 650	36
REGEL 660	49
REGEL 670	89
REGEL 680	33
REGEL 690	95
REGEL 700	175
REGEL 710	36
REGEL 720	68
REGEL 730	38
REGEL 740	183
REGEL 750	198
REGEL 760	89
REGEL 770	75
REGEL 780	180
REGEL 790	110
REGEL 800	152
REGEL 810	125
REGEL 820	63
REGEL 830	89
REGEL 840	33
REGEL 850	216
REGEL 860	245
REGEL 870	131
REGEL 880	107
REGEL 890	40
REGEL 1000	49
REGEL 1005	23
REGEL 1010	162
REGEL 1020	153
REGEL 1030	121
REGEL 1040	195
REGEL 1050	189
REGEL 1060	202
REGEL 1070	188
REGEL 1080	115
REGEL 1090	130
REGEL 1100	227
REGEL 1110	85
REGEL 1120	211
REGEL 1130	89
REGEL 1140	147
REGEL 1150	142
REGEL 1200	99
REGEL 1210	126
REGEL 1220	49
REGEL 1230	209
REGEL 1240	46
REGEL 1250	85
REGEL 1260	44
REGEL 1270	151
REGEL 1280	30
REGEL 1290	248

Programma key maze

De bedoeling van Key Maze is dat de speler 1 of meer sleutels moet verzamelen om tot de hogere levels te komen. Tijdens het spel kan de speler kleine pillen (dit zijn de kleine gele stippen) eten om punten te scoren. Ook kan de speler een krachtpil (dit zijn de grote grijze pillen) eten om de monsters te pakken. Om bij het volgende level te komen moet de speler tenminste 1 sleutel en 25 of meer kleine pillen hebben verzameld. Bij dit spel zijn er 20 levels, 3 monsters, en 2 verschillende speelvelden. De maker van al dit schoons is Y.S Kasmin uit Amsterdam.

```
1      rem********************************
       ***
2      rem**      ==key maze==
       **
3      rem**
       **
4      rem**   door: y.s.kasmin
       **
5      rem**
       **
6      rem**    amsterdam-west
       **
7      rem****————————————*
       ***
8      rem****** voor  commodore info ***
       ***
9      rem********************************
       ***
100    print"[SHIFT-CLR]":poke52,48:poke5
       6,48:clr:gosub950:gosub860
105    ap=3:vl=1:pt=0:poke53281,0:poke532
       80,0
110    sv=int(2*rnd(0)+1)
115    sl=0:on sv gosub 535,670:gosub405
120    x=b1:y=b2:s=b3:s1=b4:s2=b5:s3=b6:s
       4=b7:s5=b8:w=x:w1=y:w2=s:w3=s1:w4=
       s2:w5=s3
125    w6=s4:w7=s5:sc=1024:cc=55296:as=0:
       kn=35:sk=5:ck=2
130    rem********* hoofdlus
135    j=peek(56320):kn=kn+1:ifkn=37thenk
       n=35
140    ifan=1thenda=da+1:ifda=25thenan=0:
       sk=5:ck=2
145    dx=((j=123)-(j=119)):dy=((j=126)-(
       j=125))
150    x=x+dx:y=y+dy:b=peek(sc+x+40*y)
155    ifb=40thenx=w:y=w1
160    ifb=39thenpt=pt+5:as=as+1
165    ifb=38thenpt=pt+10:an=1:da=0:sk=2:
       ck=5:gosub490
170    ifb=41thenpt=pt+25:sl=sl+1:gosub50
       5
175    ifrnd(0)<vl/10thengosub260
180    ifx=sthenify=s1thengosub315
185    ifx=s2thenify=s3thengosub330
190    ifx=s4thenify=s5thengosub345
195    gosub405
200    v=(sv=1andas>=25andsl>=1)and(x>=20
       andx<=21andy>=22)
205    ifvthengosub375:goto115
```

```
210    v=(sv=2andas>=25andsl>=1)and(x>=4a
       ndx<=5andy>=22)
215    ifvthengosub375:goto115
220    ifap=0thenprint"[SHIFT-CLR]":goto4
       60
225    rem******* plot karakters
230    pokesc+w+40*w1,32:pokesc+w2+40*w3,
       32:pokesc+w4+40*w5,32:pokesc+w6+40
       *w7,32
235    pokesc+x+40*y,kn:pokesc+s+40*s1,37
       :pokesc+s2+40*s3,37:pokesc+s4+40*s
       5,37
240    pokecc+x+40*y,sk:pokecc+s+40*s1,ck
       :pokecc+s2+40*s3,ck:pokecc+s4+40*s
       5,ck
245    w=x:w1=y:w2=s:w3=s1:w4=s2:w5=s3:w6
       =s4:w7=s5:goto135
250    rem******* beweeg routine
255    rem******* spook 1
260    m=sgn(x-s):m1=sgn(y-s1)
265    s=s+m:s1=s1+m1:ifpeek(sc+s+40*s1)=
       40thens=w2:s1=w3
270    rem******* spook 2
275    m2=sgn(x-s2):m3=sgn(y-s3)
280    s2=s2+m2:s3=s3+m3:ifpeek(sc+s2+40*
       s3)=40thens2=w4:s3=w5
285    rem******* spook 3
290    m4=sgn(x-s4):m5=sgn(y-s5)
295    s4=s4+m4:s5=s5+m5:ifpeek(sc+s4+40*
       s5)=40thens4=w6:s5=w7
300    return
305    rem******* botst-routine
310    rem******* spook 1
315    ifan=1thenpt=pt+50:s=b3:s1=b4:goto
       520
320    goto 360
325    rem******* spook 2
330    ifan=1thenpt=pt+50:s2=b5:s3=b6:got
       o520
335    goto 360
340    rem******* spook 3
345    ifan=1thenpt=pt+50:s4=b7:s5=b8:got
       o520
350    goto 360
355    rem******* speler dood
360    gosub520
365    ap=ap-1:x=b1:y=b2:sl=0:da=0:as=0:a
       n=0:return
370    rem******* bepaal level
375    poke53281,0:vl=vl+1:ifvl=21thengot
       o420
380    pokesi+4,33:forp=0to125:poke53281,
       rnd(0)*15:pokesi+1,rnd(0)*200:next
385    pokesi+4,0:poke53281,0
390    sv=sv+1:ifsv=3thensv=1
395    return
400    rem******* status
405    print"[HOME][CRSR-DOWN][CRSR-RIGHT
       ][COM-7]level:"vl"[SPACE]levens:"a
       p"[2xCRSR-RIGHT]aantal[SPACE][CTRL
        8]'[COM-7]:"as"[CRSR-LEFT][2xSPAC
       E]"
410    print"[CRSR-RIGHT][COM-7]score:"pt
       ;tab(15)"sleutel[s]:"sl:return
415    rem* alle twintig levels gehaald
420    pokesi+4,33
425    forp=0to75:poke53281,int(15*rnd(0)
       +1):poke53280,int(7*rnd(0)+1)
430    pokesi+1,int(76*rnd(0)+70):next:po
       kesi+4,0:poke53281,2:poke53280,7:p
       rint"[SHIFT-CLR]"
435    bo=int(1000*rnd(0)+100)
440    printtab(9)"[CTRL-2][4xCRSR-DOWN]a
       lle[SPACE]levels[SPACE]gehaald!![2
       xCRSR-DOWN]":printtab(13)"bonus:"b
       o
445    pt=pt+bo
450    goto 465
455    rem**** spel einde
460    printtab(10)"[CTRL-8][4xCRSR-DOWN]
       g[SPACE]a[SPACE]m[SPACE]e[SPACE]-[
       SPACE]o[SPACE]v[SPACE]e[SPACE]r"
465    printtab(9)"[CTRL-2][2xCRSR-DOWN]j
       ouw[SPACE]score[SPACE]is:"mid$(str
       $(pt),2)
470    print"[3xCRSR-DOWN][2xCRSR-RIGHT][
       CTRL-8]druk[SPACE]op[SPACE]vuurkno
       p[SPACE]voor[SPACE]een[SPACE]nieuw
       [SPACE]spel"
475    ifpeek(56320)<>111then475
480    goto105
485    rem***  geluid voor krachtpil
490    pokesi+4,33
495    forp=10to35:pokesi+1,p:next:pokesi
       +4,0:return
500    rem******* geluid bij sleutel
505    pokesi+4,33
510    forp=15to100:pokesi+1,p:next:pokes
       i+4,0:return
515    rem******** spook/monster dood
520    pokesi+4,33
525    forp=15to175:pokesi+1,rnd(0)*p:nex
       t:pokesi+4,0:return
530    rem****** speelveld 1
535    print"[SHIFT-CLR][HOME][COM-2]((((
       ((((((((((((((((((((((((((((((((((
       ("
540    print"([37xSPACE]("
545    print"([37xSPACE]("
550    print"(((((((((((((((((((((((((((((
       ((((((((((("
555    print"([SPACE]([4xSPACE]([6xSPACE]
       ([9xSPACE]([5xSPACE](((([4xSPACE](
       "
560    print"([SPACE](((([SPACE]([4xSPACE
       ](((([9xSPACE](((([2xSPACE](((([4xS
       PACE]("
565    print"([6xSPACE]([4xSPACE]([14xSPA
       CE]([5xSPACE]((([2xSPACE]("
570    print"([SPACE](((((([2xSPACE]((([1
       4xSPACE]((([3xSPACE]((([2xSPACE]("
575    print"([SPACE]([7xSPACE]([4xSPACE]
       (((((((((([4xSPACE]([3xSPACE]((([3x
       SPACE]("
580    print"([SPACE]([SPACE]((([SPACE]([
       3xSPACE](([SPACE]([6xSPACE]([SPACE
       ]([SPACE]([4xSPACE]([SPACE]((([3xSP
       ACE]("
585    print"([SPACE]([SPACE]([3xSPACE]([
       SPACE]([SPACE]((([SPACE]([SPACE](((
       [2xSPACE]([SPACE]([SPACE]([SPACE](
       [2xSPACE]([6xSPACE]("
590    print"([3xSPACE]([SPACE]((([SPACE]
       ([SPACE]([4xSPACE]([4xSPACE]([5xSP
       ACE]([2xSPACE]((([4xSPACE]("
595    print"(((([5xSPACE]([SPACE]([SPAC
```

```
     E]([SPACE]([2xSPACE]([2xSPACE]([2x
     SPACE]([2xSPACE]([9xSPACE]("
600  print"([5xSPACE]([3xSPACE]([3xSPAC
     E]([SPACE]([2xSPACE](((([2xSPACE](
     [2xSPACE]([4xSPACE]([4xSPACE]("
605  print"([5xSPACE]([3xSPACE]((([SPAC
     E]([SPACE]([8xSPACE]([SPACE]((([4xS
     PACE]([4xSPACE]("
610  print"([3xSPACE]((([SPACE]((([3xSP
     ACE](((((((((((([2xSPACE](((([SPAC
     E]((([2xSPACE]("
615  print"([3xSPACE]([30xSPACE]([2xSPA
     CE]("
620  print"([3xSPACE]([SPACE]((([SPACE]
     (((((((((([11xSPACE](([2xSPACE]([2
     xSPACE]("
625  print"([3xSPACE]([SPACE]((([SPACE]
     (((((([6xSPACE](((((((([SPACE]([3x
     SPACE]([2xSPACE]("
630  print"([7xSPACE]([SPACE]([7xSPACE]
     (((((([5xSPACE]([SPACE]([6xSPACE](
     "
635  print"([7xSPACE]([22xSPACE]([6xSPA
     CE]("
640  print"([3xSPACE]((([SPACE]((([SPAC
     E]((((((([2xSPACE](((((([SPACE]((
     ([SPACE]((([2xSPACE]("
645  print"([3xSPACE]((([12xSPACE]([2xS
     PACE]([10xSPACE]((([2xSPACE]("
650  print"([3xSPACE](((((((([7xSPACE](
     [CTRL-5](([COM-2]([5xSPACE](((((((
     (((("
655  print"((((((((((((((((((((((((((((
     ((((((((((([HOME]"
660  b1=4:b2=4:b3=9:b4=20:b5=19:b6=4:b7
     =30:b8=20:goto805
665  rem***** speelveld 2
670  print"[SHIFT-CLR][HOME][COM-2]((((
     ((((((((((((((((((((((((((((((((((
     ("
675  print"([37xSPACE]("
680  print"([37xSPACE]("
685  print"((((((((((((((((((((((((((((
     ((((((((((("
690  print"([13xSPACE]([8xSPACE]((([SPA
     CE]([5xSPACE]([4xSPACE]("
695  print"((((((((((((([SPACE]([9xSPAC
     E]([2xSPACE]([5xSPACE]([4xSPACE]("
700  print"([11xSPACE]([SPACE]([2xSPACE
     ](((((([SPACE]([SPACE](([2xSPACE](
     ([6xSPACE]("
705  print"([SPACE]((((((((([SPACE]([SP
     ACE]([2xSPACE]([4xSPACE]([SPACE]([
     SPACE]([3xSPACE]([2xSPACE](((([SPA
     CE]("
710  print"([SPACE]([7xSPACE]([SPACE]([
     SPACE]([2xSPACE]([SPACE]([2xSPACE]
     ([SPACE]([3xSPACE]((([7xSPACE]("
715  print"([9xSPACE]([SPACE]([SPACE]([
     2xSPACE]([SPACE]([2xSPACE]([SPACE]
     ((([SPACE](((((([SPACE](((("
720  print"([SPACE]((((((((([SPACE]([SP
     ACE](((([SPACE](((([SPACE]((([SPAC
     E]([4xSPACE]([2xSPACE]((("
725  print"([SPACE]([9xSPACE]([11xSPACE
     ]((([SPACE]([SPACE]([2xSPACE]([2xS
     PACE]((("
730  print"([SPACE]([9xSPACE]([2xSPACE]
     ((([10xSPACE]([SPACE]([2xSPACE](([
     3xSPACE]("
735  print"([SPACE]((((((((((([2xSPACE]
     ([4xSPACE](((((((((([SPACE](([SPACE
     ](((([SPACE]("
740  print"([14xSPACE]([SPACE]([2xSPACE
     ]([12xSPACE](([3xSPACE]("
745  print"([12xSPACE]([SPACE]([SPACE](
     [2xSPACE]([2xSPACE](((((((((((([3x
     SPACE]("
750  print"([12xSPACE]([3xSPACE]([10xSP
     ACE]([5xSPACE](([2xSPACE]("
755  print"([7xSPACE](((([SPACE](((((((
     ((([2xSPACE](([SPACE]([9xSPACE]("
760  print"([10xSPACE](((((((((([7xSPAC
     E]((([7xSPACE]("
765  print"((((((((([15xSPACE](([2xSPAC
     E]((([7xSPACE]("
770  print"(([6xSPACE]([15xSPACE](([4xS
     PACE]([5xSPACE]([SPACE]("
775  print"(([SPACE]([2xSPACE]([SPACE](
     [SPACE](((((((((((([SPACE]((((([SP
     ACE](([5xSPACE]([SPACE]("
780  print"([2xSPACE]([2xSPACE]([SPACE]
     ([9xSPACE](((([SPACE]((((([6xSPACE
     ]((([SPACE]("
785  print"([2xSPACE]([CTRL-5](([COM-2]
     ([5xSPACE]((((([21xSPACE]("
790  print"((((((((((((((((((((((((((((
     ((((((((((([HOME]"
795  b1=3:b2=12:b3=6:b4=18:b5=18:b6=16:
     b7=35:b8=21
800  rem***** plot pil/krachtpil/sleute
     l
805  for r=0to50
810  x=int(37*rnd(0)+1):y=int(20*rnd(0)
     +4):ifpeek(1024+x+40*y)<>32then810
815  poke55296+x+40*y,7:poke1024+x+40*y
     ,39:next
820  forr=0to5
825  x=int(37*rnd(0)+1):y=int(20*rnd(0)
     +4)
830  ifpeek(1024+x+40*y)<>32then825
835  poke1024+x+40*y,41:poke55296+x+40*
     y,3:next
840  for r=0to6
845  x=int(37*rnd(0)+1):y=int(20*rnd(0)
     +4):ifpeek(1024+x+40*y)<>32then845
850  poke1024+x+40*y,38:poke55296+x+40*
     y,12:next:return
855  rem****titelscherm
860  printchr$(8)chr$(142)
865  poke53281,0:poke53280,0:print"[SHI
     FT CLR][HOME]";
870  printtab(15)"[CRSR-DOWN][COM-1](((
     ((((((("
875  print"[CRSR-DOWN][7xCRSR-RIGHT](((
     ((((([SPACE][CTRL-8]key)maze[COM-1
     ][SPACE](((((((((":printtab(15)"[CR
     SR-DOWN](((((((((("
880  printtab(11)"[2xCRSR-DOWN][CTRL-2]
     door:[SPACE]yerrel[SPACE]kasmin"
885  print"[3xCRSR-DOWN][CTRL-6][7xCRSR
     -RIGHT]#...[COM-6]speler[7xSPACE][
     CTRL-3]%...[COM-6]spook"
890  print"[CRSR-DOWN][7xCRSR-RIGHT][CO
     M 8]&...[COM-6]krachtpil[4xSPACE][
     CTRL-8]'...[COM-6]pil"
```

```
895   printtab(14)"[CRSR-DOWN][CTRL-4]).
      ..[COM-6]sleutel"
900   printtab(12)"[2xCRSR-DOWN][COM-3]#
      [SPACE]druk[SPACE]op[SPACE]vuurkno
      p[SPACE]#"
905   for x=0to20
910   r=int(990*rnd(0)+1):ifpeek(1024+r)
      <>32then910
915   poke1024+r,39:poke55296+r,4:next:d
      x=1:x=0:bx=x:cn=35
920   poke56216+x,5:poke1944+bx,32:poke1
      944+x,cn
925   bx=x:ifpeek(56320)=111thenreturn
930   x=x+dx:ifx<=0orx>=39thendx=-dx
935   cn=cn+1:ifcn=37thencn=35
940   forp=0to175:next:goto920
945   rem*** init. ******
950   printtab(215)"geduld......"
955   si=54272:forx=sitosi+24:pokex,0:ne
      xt:pokesi+0,100::pokesi+6,250:poke
      si+24,15
960   poke56334,peek(56334)and254:poke1,
      peek(1)and251
965   forx=0to511:poke12288+x,peek(53248
      +x):nextx
970   poke1,peek(1)or4:poke56334,peek(56
      334)or1:poke53272,(peek(53272)and2
      40)+12
975   forka=35to41:forby=0to7:readd
980   poke12288+(8*ka)+by,d:nextby,ka:re
      turn
985   data 129,66,60,90,255,189,66,60
990   data 0,195,60,90,255,255,66,60
995   data 60,90,126,231,90,60,90,153
1000  data 0,60,110,94,94,94,60,0
1005  data 0,0,24,60,60,24,0,0
1010  data 231,195,165,24,24,165,195,231
1015  data 0,2,37,45,125,7,2,0
```

EINDE LISTING key maze

Checksum Key Maze

REGEL 1	35	REGEL 310	114	REGEL 565	29	REGEL 820	142
REGEL 2	65	REGEL 315	161	REGEL 570	133	REGEL 825	176
REGEL 3	55	REGEL 320	34	REGEL 575	133	REGEL 830	137
REGEL 4	112	REGEL 325	115	REGEL 580	93	REGEL 835	16
REGEL 5	80	REGEL 330	217	REGEL 585	133	REGEL 840	143
REGEL 6	165	REGEL 335	34	REGEL 590	13	REGEL 845	117
REGEL 7	113	REGEL 340	116	REGEL 595	13	REGEL 850	14
REGEL 8	158	REGEL 345	225	REGEL 600	229	REGEL 855	123
REGEL 9	35	REGEL 350	34	REGEL 605	189	REGEL 860	152
REGEL 100	180	REGEL 355	166	REGEL 610	21	REGEL 865	29
REGEL 105	84	REGEL 360	36	REGEL 615	125	REGEL 870	49
REGEL 110	86	REGEL 365	159	REGEL 620	213	REGEL 875	129
REGEL 115	72	REGEL 370	210	REGEL 625	77	REGEL 880	54
REGEL 120	160	REGEL 375	67	REGEL 630	189	REGEL 885	250
REGEL 125	150	REGEL 380	196	REGEL 635	125	REGEL 890	12
REGEL 130	109	REGEL 385	157	REGEL 640	61	REGEL 895	40
REGEL 135	220	REGEL 390	101	REGEL 645	109	REGEL 900	105
REGEL 140	33	REGEL 395	142	REGEL 650	206	REGEL 905	193
REGEL 145	101	REGEL 400	153	REGEL 655	8	REGEL 910	20
REGEL 150	121	REGEL 405	37	REGEL 660	160	REGEL 915	214
REGEL 155	184	REGEL 410	105	REGEL 665	55	REGEL 920	216
REGEL 160	90	REGEL 415	174	REGEL 670	48	REGEL 925	246
REGEL 165	178	REGEL 420	163	REGEL 675	45	REGEL 930	109
REGEL 170	252	REGEL 425	241	REGEL 680	45	REGEL 935	115
REGEL 175	246	REGEL 430	27	REGEL 685	245	REGEL 940	14
REGEL 180	118	REGEL 435	45	REGEL 690	29	REGEL 945	107
REGEL 185	167	REGEL 440	247	REGEL 695	173	REGEL 950	10
REGEL 190	177	REGEL 445	53	REGEL 700	53	REGEL 955	161
REGEL 195	38	REGEL 450	40	REGEL 705	77	REGEL 960	227
REGEL 200	143	REGEL 455	208	REGEL 710	229	REGEL 965	41
REGEL 205	14	REGEL 460	111	REGEL 715	253	REGEL 970	208
REGEL 210	52	REGEL 465	210	REGEL 720	21	REGEL 975	72
REGEL 215	14	REGEL 470	33	REGEL 725	189	REGEL 980	147
REGEL 220	114	REGEL 475	220	REGEL 730	149	REGEL 985	158
REGEL 225	156	REGEL 480	31	REGEL 735	101	REGEL 990	95
REGEL 230	90	REGEL 485	175	REGEL 740	245	REGEL 995	134
REGEL 235	27	REGEL 490	163	REGEL 745	173	REGEL 1000	188
REGEL 240	76	REGEL 495	223	REGEL 750	245	REGEL 1005	15
REGEL 245	72	REGEL 500	98	REGEL 755	213	REGEL 1010	37
REGEL 250	138	REGEL 505	163	REGEL 760	53	REGEL 1015	29
REGEL 255	114	REGEL 510	13	REGEL 765	53		
REGEL 260	81	REGEL 515	232	REGEL 770	29		
REGEL 265	89	REGEL 520	163	REGEL 775	237		
REGEL 270	115	REGEL 525	1	REGEL 780	133		
REGEL 275	185	REGEL 530	96	REGEL 785	198		
REGEL 280	97	REGEL 535	48	REGEL 790	8		
REGEL 285	116	REGEL 540	45	REGEL 795	170		
REGEL 290	193	REGEL 545	45	REGEL 800	163		
REGEL 295	121	REGEL 550	245	REGEL 805	190		
REGEL 300	142	REGEL 555	109	REGEL 810	109		
REGEL 305	148	REGEL 560	173	REGEL 815	27		

Programma krantkolommer/64

Roalt Aalmoes uit schagen vond dat er te weinig mogelijkheden waren om goed te kunnen printen met de commodore 64. Na veel denken, passen en meten kwam de volgende listing uit zijn handen (computer).

```
1     rem de krantkolommer
2     rem door roalt aaloes van pfc
3     rem schagen
4     rem
5     rem ————————
10    gosub 50
20    gosub 330
30    gosub 360
40    goto 760
50    open4,4:print#4
60    poke53280,1:poke53281,1:poke646,0
70    printchr$(147)
80    printchr$(14)chr$(8)"[SPACE]dit[SP
      ACE]is[SPACE]de[SPACE]krant-kolomm
      er[SPACE]van[SPACE]de[SPACE]pfc"
90    print"[CRSR-DOWN][SPACE]geschikt[S
      PACE]voor[SPACE]elke[SPACE]printer
      ...[2xSPACE]"
100   print"[2xCRSR-DOWN][SPACE](tip:bij
      [SPACE]invoermode[SPACE]:'[PIJL LI
       NKS]'=regel[SPACE]terug,[8xSPACE]'
       '=eind
110   print"[7xSPACE]<space>+<return>=re
      gel[SPACE]overslaan)"
120   print"[3xCRSR-DOWN][SPACE]hoeveel[
      SPACE]kolommen[SPACE]wilt[SPACE]u[
      SPACE]?"
130   geta$:ifa$=""then130
140   kl=val(a$):ifkl<1andkl>10then130
150   print"[CRSR-DOWN][SPACE]hoeveel[SP
      ACE]tekens[SPACE]per[SPACE]kolom[S
      PACE]wilt[SPACE]u[SPACE]?";
160   inputa$:tk=val(a$)
170   if tk<5 or tk>36 then 150
180   print"[CRSR-DOWN][SPACE]hoeveel[SP
      ACE]regels[SPACE]per[SPACE]kolom[S
      PACE]wilt[SPACE]u[SPACE]?";
190   input a$
200   rk=val(a$):if rk<3 or rk>150then1
      80
210   print"[3xCRSR-DOWN][SPACE]beschikt
      [SPACE]u[SPACE]over[SPACE]een[SPAC
      E]"tk*kl+kl*5+3
220   print"printer[SPACE]?(j/n)
230   get a$:if a$=""then230
240   if a$="j"then 260
250   run
260   if tk*kl+kl*5+3>255thenrun
270   dim te$(rk*kl+1),tk$(rk+1)
280   input"[SPACE]printer[SPACE]command
      o";a$
290   a=int(val(a$)):if a<0ora>255then2
      80
300   print#4,chr$(a)
310   print"[CRSR-DOWN][SPACE]druk[SPACE
      ]op[SPACE]een[SPACE]toets"
320   return
330   poke198,0
340   wait198,1:poke198,0
350   return
360   print"[SHIFT-CLR][2xCRSR-DOWN]";
370   ei=0
380   rl=1
390   gosub1170:gosub 930:if a=95 then 1
      140
400   if rl>rk*kl-1 and ei<1then 390
410   if len(te$(rl)+tt$)<tk+1thente$(rl
      )=te$(rl)+tt$:goto440
420   if len(te$(rl)+tt$)=tk+1then470
430   gosub 1080:te$(rl)=tt$
440   printtt$;:ifei=1thengosub 1080:te$
      (rl)="":return
450   if a=13 thengosub1080:tt$=tt$+"[SP
      ACE]"
460   goto390
470   te$(rl)=te$(rl)+left$(tt$,len(tt$)
      -1):goto440
480   print"[SHIFT-CLR][2xSPACE]tekst[SP
      ACE]lezen[SPACE]...
490   print"[2xCRSR-DOWN][SPACE]<space>[
      SPACE]=[SPACE]verder[SPACE]gaan
500   print"[2xCRSR-DOWN]
510   print"[4xCRSR-DOWN][2xSPACE]druk[S
      PACE]op[SPACE]een[SPACE]toets[SPAC
      E]om[SPACE]te[SPACE]beginnen"
520   gosub330
530   rl=1
540   printrl":":printtab(4);te$(rl)
550   if te$(rl)=""then 580
560   gosub 330
570   rl=rl+1:goto 540
580   print"[CRSR-DOWN][SPACE]einde[SPAC
      E]tekst[SPACE]"
590   print"[2xCRSR-DOWN][SPACE]nog[SPAC
      E]eens[SPACE]lezen[SPACE]?[SPACE](
      j/n)"
600   geta$:ifa$=""then 600
610   if a$="n"thenreturn
620   ifa$="j"then480
630   goto600
640   sp$="[60xSPACE]":vs$="[2xSPACE]"
650   sp$=left$(sp$,tk+5)
660   print"[SHIFT-CLR][SPACE]koloms[SPA
      CE]maken[SPACE]..."
670   for nr=1tork
680   tp$(nr)=vs$+te$(nr)+left$(sp$,len(
      sp$)-len(te$(nr)))
690   if kl<2thennextnr:goto730
700   fori=1tokl-1
710   tp$(nr)=tp$(nr)+te$(nr+rk*i)+left$
      (sp$,len(sp$)-len(te$(nr+rk*i)))
720   nexti:nextnr
730   print"[4xCRSR-DOWN][SPACE]koloms[S
      PACE]gemaakt...[3xCRSR-DOWN]druk[S
      PACE]op[SPACE]een[SPACE]toets"
740   gosub330
750   return
760   print"[SHIFT-CLR][10xSPACE]keuze-m
      enu
770   print"[3xCRSR-DOWN][SPACE]1][SPACE
      ]tekst[SPACE]lezen
780   print"[2xCRSR-DOWN][SPACE]2][SPACE
      ]tekst[SPACE]kolommeren[SPACE]en[S
      PACE]afdrukken
790   print"[2xCRSR-DOWN][SPACE]3][SPACE
      ]nieuwe[SPACE]tekst[SPACE]invoeren
800   print"[2xCRSR-DOWN][SPACE]4][SPACE
      ]terug[SPACE]naar[SPACE]basic
```

```
810   print"[4xCRSR-DOWN][SPACE]maak[SPA
      CE]uw[SPACE]keuze
820   get a$:ifa$=""then820
830   if a$="1"thengosub480:goto760
840   if a$="2"then870
850   if a$="3"then print#4:close4:run
860   if a$="4"thensys64738
870   gosub 640:print"[SHIFT-CLR][CRSR-D
      OWN][SPACE]printen[SPACE]..."
880   print#4,"[SPACE]de[SPACE]krant[SPA
      CE]kolommer[SPACE]pfc":print#4:pri
      nt#4:print#4:fora=1tork
890   print#4,tp$(a):print"[HOME]regel:"
      a
900   nexta
910   print#4:print#4:print#4:print#4
920   goto760
930   tt$="":g=0:fora=0to36:poke1024+a,3
      2:next
940   geta$:ifa$=""then 940
950   a=asc(a$)
960   if a$="`"thenei=1:goto1060
970   if a$="[PIJL LINKS]"thenreturn
980   ifa=20andlen(tt$)>0thentt$=left$(t
      t$,len(tt$)-1):g=g-1:pokeg+1024,32
990   ifa=20then940
1000  ifa=13andlen(tt$)>0then1060
1010  ifa<32 then940
1020  ifa>127 and a<161 then940
1030  ifa=32andlen(tt$)>0thentt$=tt$+"[S
      PACE]":goto1060
1040  iflen(tt$)<tk*kl+1thentt$=tt$+a$:g
      osub1100:g=g+1
1050  goto940
1060  if len(tt$)>tk*kl+1thentt$=left$(t
      t$,len(tt$)-1):goto940
1070  return
1080  rl=rl+1:print
1090  return
1100  if a>64anda<96thena=a-64:goto1120
1110  ifa>191thena=a-128:goto1120
1120  pokeg+1024,a
1130  return
1140  print"[CRSR-UP]":print"[39xSPACE]"
      :print"[CRSR-UP]";
1150  if te$(rl)=""andrl>1thenrl=rl-1:pr
      int"[CRSR-UP]";:goto1140
1160  te$(rl)="":goto390
1170  rl$=str$(rl):poke1063,32:poke1062,
      32:poke1061,32:poke1060,18
1180  poke1063,val(right$(rl$,1))+48:ifr
      l<10thenreturn
1190  rl$=left$(rl$,len(rl$)-1)
1200  poke1062,val(right$(rl$,1))+48:ifr
      l<100thenreturn
1210  rl$=left$(rl$,len(rl$)-1)
1220  poke1061,val(right$(rl$,1))+48:ret
      urn
```

EINDE LISTING krantkolommer/64

Checksum Krantkolommer

REGEL 1	254	REGEL 360	205	REGEL 760	52	REGEL 1160	1
REGEL 2	184	REGEL 370	112	REGEL 770	133	REGEL 1170	213
REGEL 3	136	REGEL 380	129	REGEL 780	30	REGEL 1180	158
REGEL 4	185	REGEL 390	76	REGEL 790	43	REGEL 1190	45
REGEL 5	185	REGEL 400	250	REGEL 800	121	REGEL 1200	205
REGEL 10	242	REGEL 410	112	REGEL 810	73	REGEL 1210	45
REGEL 20	35	REGEL 420	47	REGEL 820	103	REGEL 1220	18t
REGEL 30	38	REGEL 430	186	REGEL 830	71		
REGEL 40	38	REGEL 440	119	REGEL 840	94		
REGEL 50	57	REGEL 450	81	REGEL 850	94		
REGEL 60	244	REGEL 460	37	REGEL 860	107		
REGEL 70	77	REGEL 470	220	REGEL 870	140		
REGEL 80	10	REGEL 480	225	REGEL 880	182		
REGEL 90	85	REGEL 490	223	REGEL 890	102		
REGEL 100	42	REGEL 500	221	REGEL 900	195		
REGEL 110	200	REGEL 510	216	REGEL 910	222		
REGEL 120	78	REGEL 520	35	REGEL 920	38		
REGEL 130	97	REGEL 530	129	REGEL 930	224		
REGEL 140	151	REGEL 540	111	REGEL 940	106		
REGEL 150	56	REGEL 550	113	REGEL 950	111		
REGEL 160	240	REGEL 560	35	REGEL 960	200		
REGEL 170	184	REGEL 570	37	REGEL 970	122		
REGEL 180	48	REGEL 580	222	REGEL 980	178		
REGEL 190	234	REGEL 590	227	REGEL 990	36		
REGEL 200	230	REGEL 600	99	REGEL 1000	192		
REGEL 210	167	REGEL 610	105	REGEL 1010	40		
REGEL 220	54	REGEL 620	115	REGEL 1020	150		
REGEL 230	98	REGEL 630	31	REGEL 1030	188		
REGEL 240	111	REGEL 640	186	REGEL 1040	185		
REGEL 250	138	REGEL 650	3	REGEL 1050	38		
REGEL 260	234	REGEL 660	59	REGEL 1060	43		
REGEL 270	7	REGEL 670	69	REGEL 1070	142		
REGEL 280	219	REGEL 680	142	REGEL 1080	156		
REGEL 290	220	REGEL 690	45	REGEL 1090	142		
REGEL 300	81	REGEL 700	196	REGEL 1100	112		
REGEL 310	42	REGEL 710	242	REGEL 1110	192		
REGEL 320	142	REGEL 720	39	REGEL 1120	188		
REGEL 330	149	REGEL 730	233	REGEL 1130	142		
REGEL 340	96	REGEL 740	35	REGEL 1140	104		
REGEL 350	142	REGEL 750	142	REGEL 1150	57		

Programma ruimte landing

De maker van dit spel Y.S Kasmin. De bedoeling van dit spel is om een ruimteschip veilig te laten landen. Het schip moet op het groene plekje landen met een snelheid dat niet hoger dan 20 mag zijn. Ook geldt dit voor de X positie van het ruimteschip, X moet tussen de 140-en de 150 liggen. Voldoet de landing niet aan deze eisen dan volgt er een explosie waarbij de speler een schip verliest. Voldoet de landing wel aan deze eisen, dan gaat de speler naar een hoger level. De speler kan de X positie tijdens de landing zien door op de vuurknop te drukken, dit duurt ongeveer 9 seconden waarna het verdwijnt. Deze functie kan alleen zes keer per 3 seconden worden gebruikt. Deze functie staat boven op het beeldscherm aangegeven in INFO. Is de speler 3 keer goed geland dan krijgt de speler bonus punten toegekend plus 6 nieuwe INFO's en het spelniveau wordt verhoogt. Dit spel heeft 1 tegenstander die sneller gaat achtervolgen bij elke level verandering, plus willekeurige rukwinden die ook heviger worden Dit spel heeft geen level grens.

```
1     rem ****************************
2     rem *                          *
3     rem *     ruimte landing       *
4     rem *     --------------     *
5     rem *          door            *
6     rem *     yerrel kasmin        *
7     rem *                          *
8     rem * voor: commodore-info     *
9     rem *                          *
10    rem****************************
100   printchr$(142):gosub405:hi=1000:go
      to460
105   lv=1:pu=0:sh=3:ag=0:in=6:poke53281
      ,0:poke53280,0
110   print"[SHIFT-CLR]":gosub 385
115   pokev+30,0:ifsh=0then345
120   rem**** begin-positie initialisere
      n
125   x=int(205*rnd(0)+50):y=50:x1=25:y1
      =50:dy=0
130   pokev,x:pokev+1,y:pokev+2,x1:pokev
      +3,y1:pokev+21,3:pokes+4,129
135   rem******* hoofdlus *********
140   ifrnd(0)>(lv/10)then160
145   d1=sgn(x-x1)*2:d2=sgn(y-y1)*2
150   x1=x1+d1:y1=y1+d2
155   pokev+2,x1:pokev+3,y1
160   j=peek(56320):fr=jand16:j=15-(jand
      15)
165   iffr=0andin>=1thentc=1:ct=0:in=in-
      1
170   iftc=1thenct=ct+1:printin$tab(11)"
      [CTRL-9][CTRL-4][SPACE]x-positie:"
      x"[CRSR-LEFT][SPACE]"
175   ifct=30thentc=0:printin$tab(11)wx$
180   ifpeek(v+30)=3thengosub310:goto115
185   ifj=2thendy=dy+.2:pokes+1,2
190   ifj=1thendy=dy-.07:pokes+1,4
195   ifj=0thendy=dy+.08:pokes+1,3
200   ifrnd(0)>(lv/10)then220
205   r=int(rnd(0)*2)+1
210   ifr=1andx>50thenx=x-5
215   ifr=2andx<250thenx=x+5
220   ifj=4andx>50thenx=x-4
225   ifj=8andx<250thenx=x+4
230   printchr$(19)"[CTRL-2]score:"pu;ta
      b(17)"info:"in;tab(29)"level:"lv:p
      rint
235   print"ruimteschip:"sh;tab(27)"snel
      heid:"int(dy*10)"[CRSR-LEFT][SPACE
      ]"
240   y=y+dy
245   poke53248,x:poke53249,y
250   ifint(y)>=185thengosub265:on(vl=1)
      +2goto110:goto115
255   goto 140
260   rem****** goede landing
265   ct=30:tc=0:ifx<140orx>150orint(dy*
      10)>20then310
270   rx=x:pokev+5,205:pokev+21,7:pokes+
      4,33
275   fori=rxto0step-1:forp=40to30step-2
      :pokes+1,p:pokev+4,i:nextp,i:pokes
      +4,0
280   pu=pu+25:lv=lv+1:ag=ag+1:vl=0
285   ifag<>3thenreturn
290   bo=int(rnd(0)*500):pu=pu+bo:print"
      [SHIFT-CLR]":pokev+21,0
295   print"[4xCRSR-DOWN][7xCRSR-RIGHT]j
      e[SPACE]bent[SPACE]3[SPACE]keer[SP
      ACE]goed[SPACE]geland"
300   printtab(12)"[CRSR-DOWN]bonus:"bo"
      punten.":ag=0:vl=1:in=6
305   forp=0to2500:next:return
310   rem**** slechte landing!!!!
315   ct=30:tc=0:pokev+21,0:vl=0:fori=0t
      o50
320   poke53270,(peek(53270)and248)+7
325   poke53265,(peek(53265)and248)+7
330   pokes+1,int(rnd(0)*25)+1:poke53265
      ,27:poke53270,200:nexti
335   pokes+4,0:pokev+21,0
340   sh=sh-1:return
345   rem***** game over
350   poke53281,9:poke53280,8:print"[SHI
      FT CLR]"
355   printtab(10)"[CTRL-2]g[SPACE]a[SPA
      CE]m[SPACE]e[SPACE]-[SPACE]o[SPACE
      ]v[SPACE]e[SPACE]r":print:print
360   ifpu>hithenhi=pu
365   print"[2xCRSR-RIGHT][CTRL-1]je[SPA
      CE]hebt[SPACE]'t[SPACE]volgehouden
      [SPACE]tot[SPACE]level:"lv:print:p
      rint:print
370   pu$=mid$(str$(pu),2):hi$=mid$(str$
      (hi),2)
375   print"[2xCRSR-RIGHT][CTRL-8]score:
      "pu$;spc(9)"[SPACE]high[SPACE]scor
      e:"hi$:goto460
380   rem****** teken speelveld
385   fori=1904to2023:pokei,160:pokei+54
      272,2:next
390   fori=56191to56197:pokei,5:next
395   fori=0to50:r=int(rnd(0)*879):poke1
      024+r,46:poke55296+r,1:next:return
400   rem****** init.
405   print"[SHIFT-CLR]":poke53281,0:pok
      e53280,0
```

```
410   fori=0to62:readd:poke12288+i,d:nex
      t
415   fori=0to62:readd:poke12352+i,d:nex
      t
420   fori=0to62:readd:poke12416+i,d:nex
      t
425   poke2040,192:poke2041,193:poke2042
      ,194:s=54272:v=53248
430   fori=stos+24:pokei,0:next:pokes,10
      0:pokes+6,240:pokes+24,15
435   pokev+23,1:pokev+29,1:pokev+39,3:p
      okev+40,8:pokev+41,7
440   in$="[HOME][23xCRSR-DOWN]":rem***
      in$=[home] [23xcrsr-down]
445   wx$="[CTRL-9][CTRL-3][16xSPACE]":r
      em**** wx$=[ctrl-9] [ctrl-3] [16xs
      pace]
450   return
455   rem****** titel
460   printtab(247)"r[SPACE]u[SPACE]i[SP
      ACE]m[SPACE]t[SPACE]e[SPACE]-[SPA
      CE]l[SPACE]a[SPACE]n[SPACE]d[SPACE
      ]i[SPACE]n[SPACE]g"
465   printtab(58)"door"tab(93)"yerrel[S
      PACE]kasmin"
470   printtab(92)"druk[SPACE]op[SPACE]v
      uurknop":wait56320,16,16:goto105
475   rem******** sprite data's
480   rem***** ruimteschip
485   data 32,127,4,32,8,4,33
490   data 201,196,35,255,228,38,0
495   data 52,47,129,252,63,129,252
500   data 60,60,62,111,255,247,127
505   data 255,255,106,126,87,127,255
510   data 255,64,60,1,127,255,255
515   data 127,1,255,63,255,254,3
520   data 129,224,3,255,224,3,34
525   data 112,127,65,127,127,128,255
530   rem***** data insektoide
535   data 1,0,128,0,129,0,0
540   data 126,0,0,219,0,0,36
545   data 0,0,24,0,3,255,192
550   data 7,255,224,44,0,52,25
555   data 255,152,17,255,136,32,0
560   data 4,0,255,0,0,255,0
565   data 0,0,0,0,126,0,0
570   data 126,0,0,0,0,0,60
575   data 0,0,24,0,0,0,0
580   rem***** data robot
585   data 0,15,1,0,63,193,0
590   data 255,241,0,7,249,0,7
595   data 249,0,255,241,0,15,1
600   data 0,255,243,105,255,250,159
605   data 0,94,152,0,92,79,255
610   data 156,9,0,56,1,255,248
615   data 0,255,240,0,31,128,0
620   data 32,64,0,255,240,1,182
625   data 216,1,182,216,0,255,240
```

EINDE LISTING ruimte landing

Checksum Ruimtelanding

REGEL 1	39	REGEL 350	224
REGEL 2	227	REGEL 355	56
REGEL 3	182	REGEL 360	1
REGEL 4	89	REGEL 365	216
REGEL 5	23	REGEL 370	110
REGEL 6	121	REGEL 375	163
REGEL 7	227	REGEL 380	166
REGEL 8	97	REGEL 385	116
REGEL 9	227	REGEL 390	105
REGEL 10	39	REGEL 395	108
REGEL 100	67	REGEL 400	237
REGEL 105	188	REGEL 405	207
REGEL 110	215	REGEL 410	178
REGEL 115	219	REGEL 415	170
REGEL 120	182	REGEL 420	171
REGEL 125	184	REGEL 425	116
REGEL 130	199	REGEL 430	117
REGEL 135	147	REGEL 435	192
REGEL 140	183	REGEL 440	127
REGEL 145	105	REGEL 445	251
REGEL 150	3	REGEL 450	142
REGEL 155	56	REGEL 455	13
REGEL 160	234	REGEL 460	115
REGEL 165	169	REGEL 465	92
REGEL 170	45	REGEL 470	190
REGEL 175	230	REGEL 475	74
REGEL 180	8	REGEL 480	174
REGEL 185	179	REGEL 485	245
REGEL 190	234	REGEL 490	249
REGEL 195	232	REGEL 495	48
REGEL 200	180	REGEL 500	37
REGEL 205	255	REGEL 505	152
REGEL 210	198	REGEL 510	250
REGEL 215	250	REGEL 515	197
REGEL 220	192	REGEL 520	192
REGEL 225	247	REGEL 525	143
REGEL 230	76	REGEL 530	106
REGEL 235	202	REGEL 535	179
REGEL 240	171	REGEL 540	233
REGEL 245	126	REGEL 545	236
REGEL 250	174	REGEL 550	92
REGEL 255	30	REGEL 555	242
REGEL 260	236	REGEL 560	183
REGEL 265	193	REGEL 565	68
REGEL 270	168	REGEL 570	122
REGEL 275	1	REGEL 575	17
REGEL 280	173	REGEL 580	1
REGEL 285	223	REGEL 585	184
REGEL 290	113	REGEL 590	43
REGEL 295	200	REGEL 595	84
REGEL 300	215	REGEL 600	88
REGEL 305	162	REGEL 605	103
REGEL 310	192	REGEL 610	102
REGEL 315	156	REGEL 615	76
REGEL 320	87	REGEL 620	136
REGEL 325	95	REGEL 625	235
REGEL 330	33		
REGEL 335	180		
REGEL 340	140		
REGEL 345	183		

Programma sup.write 757

Met dit programma is het, o wat een wonder, mogelijk teksten te schrijven. Niet gewoon, maat alles wat wordt getypt is weg te schrijven naar een diskette. Deze tekst is later weer te laden en verschijnt dan weer exact op dezelfde wijze als zij is ingegeven weer op het beeldscherm. Inclusief alle eventueel uitgevoerde correcties. Een programma dus dat uitnodigt om te gaan experimenteren.

```
10    rem ************************
20    rem ***  super-writer 64  ***
30    rem ************************
40    rem ***  geschreven door  ***
50    rem ***  jim v/d heyden.  ***
60    rem ***                   ***
70    rem ***       breda       ***
80    rem ************************
90    print"[SHIFT-CLR]data[SPACE]lezen[
      SPACE]kost[SPACE]even[SPACE]tijd[S
      PACE]in[SPACE]basic"
100   forq=0to496
110   reada:poke49152+q,a
120   c=c+a:next
130   ifc<>62360thenprint"fout[SPACE]in[
      SPACE]data":end
140   sys49152
1000  data169,128,141,138,2,169,0,141,32
      ,208,141,33,208,141,255,2
1001  data141,0,3,169,32,133,177,162,0,1
      34,176,232,134,150,169,7
1002  data141,134,2,32,68,229,169,128,16
      0,193,32,30,171,32,228,255
1003  data240,251,201,81,240,28,201,49,1
      44,243,201,53,176,239,56
1004  data233,49,10,170,189,85,192,141,8
      0,192,189,86,192,141,81,192
1005  data76,93,192,76,226,252,93,192,15
      0,192,224,192,3,193,32,68
1006  data229,32,201,192,32,228,255,240,
      248,164,150,208,2,230,177
1007  data145,176,230,150,72,32,210,255,
      104,201,3,240,14,76,96,192
1008  data162,255,160,25,202,208,253,136
      ,208,250,96,165,150,141,255
1009  data2,165,177,141,0,3,76,19,192,32
      ,68,229,164,150,208,2,230
1010  data177,177,176,201,3,240,28,32,21
      0,255,32,201,192,230,150
1011  data160,255,162,5,136,208,253,202,
      208,250,32,228,255,201,3
1012  data208,219,76,19,192,32,228,255,2
      40,251,76,19,192,164,211
1013  data177,209,9,128,145,209,32,126,1
      92,164,211,177,209,41,127
1014  data145,209,32,126,192,96,32,92,19
      3,162,8,160,1,32,186,255
1015  data165,151,162,240,160,193,32,189
      ,255,169,0,32,213,255,169,2
1016  data32,195,255,32,231,255,76,19,19
      2,173,0,3,208,3,76,19,192
1017  data32,92,193,160,1,169,2,162,8,32
      ,186,255,165,151,162,240
1018  data160,193,32,189,255,32,192,255,
      162,2,32,201,255,169,0,133
1019  data251,32,210,255,169,32,133,252,
      32,210,255,160,0,177,251,32
1020  data210,255,230,251,208,2,230,252,
      165,252,205,0,3,208,236,165
1021  data251,205,255,2,208,229,169,2,32
      ,195,255,32,231,255,76,19
1022  data192,32,68,229,169,0,133,151,16
      9,224,160,193,32,30,171,32
1023  data228,255,240,251,72,32,210,255,
      104,166,151,157,240,193,230
1024  data151,201,13,208,235,96,49,32,78
      ,73,69,85,87,69,32,84,69,75
1025  data83,84,32,77,65,75,69,78,13,50,
      32,79,85,68,69,32,84,69,75
1026  data83,84,32,76,65,84,69,78,32,90,
      73,69,78,13,51,32,76,65,68
1027  data69,78,13,52,32,83,65,86,69,78,
      13,81,32,81,85,73,84,13,82
1028  data85,78,47,83,84,79,80,32,61,32,
      84,69,82,85,71,32,78,65,65
1029  data82,32,77,69,78,85,13,0,84,89,8
      0,69,32,70,73,76,69,78,65
1030  data65,77,13,13,0,0
```

EINDE LISTING sup.write 757

Checksum Superwrite 757

REGEL 10	169	REGEL 1012	1
REGEL 20	142	REGEL 1013	48
REGEL 30	169	REGEL 1014	251
REGEL 40	169	REGEL 1015	137
REGEL 50	31	REGEL 1016	43
REGEL 60	111	REGEL 1017	241
REGEL 70	233	REGEL 1018	81
REGEL 80	169	REGEL 1019	111
REGEL 90	155	REGEL 1020	109
REGEL 100	251	REGEL 1021	43
REGEL 110	6	REGEL 1022	83
REGEL 120	223	REGEL 1023	124
REGEL 130	96	REGEL 1024	177
REGEL 140	163	REGEL 1025	142
REGEL 1000	21	REGEL 1026	132
REGEL 1001	230	REGEL 1027	119
REGEL 1002	84	REGEL 1028	134
REGEL 1003	233	REGEL 1029	87
REGEL 1004	161	REGEL 1030	96
REGEL 1005	61		
REGEL 1006	30		
REGEL 1007	69		
REGEL 1008	132		
REGEL 1009	30		
REGEL 1010	224		
REGEL 1011	226		

Programma darts

Het programma darts heeft tot functie het tellen van de score bij een Dart spel te vereenvoudigen. U Krijgt een duidelijke uitleg van het DARTS spel. Zo kan men lezen hoe een bord opgehangen moet worden hoe de puntentelling gaat en de spelregels moet worden uitgelegd. Er kunnen max 4 spelers worden ingevoerd. De inzender en schrijver van dit bestand is S.G. Lageman uit Amsterdam.

```
1     rem darts program
2     print"[SHIFT-CLR][2xCRSR-DOWN]comm
       odore[SPACE]64[2xSPACE]darts[SPACE
       ]program[2xCRSR-DOWN][13xSPACE]voo
       r[SPACE]1[SPACE]tot[SPACE]4[SPACE]
       spelers[2xCRSR-DOWN]"
3      input"zijn[SPACE]de[SPACE]spelrege
       ls[SPACE]bekend[SPACE]j/n";i$:prin
      t:if i$="n"then23
4     print"[SHIFT-CLR][5xSPACE]spel[SPA
      CE]menu":print
5      input"hoeveel[SPACE]spelers";n:ifn
      >4 then goto 5
6     forx=1ton:print"[CRSR-DOWN]naam[SP
       ACE]speler[SPACE]"x;:input n$(x):n
      ext
```

```
7     print"[CRSR-DOWN]maak[SPACE]keuze[
       SPACE]tussen[SPACE]301[SPACE]of[SP
       ACE]501";:inputs
8     print"[SHIFT-CLR]":forx=1ton:print
      n$(x),:next:print
9      forx=1ton:s(x)=s:print"[SPACE]"s,:
       next:print
10    forx=1ton:inputsc(x)
11     ts(x)=s(x)-sc(x):ifts(x)=0thenprin
       t"[2xCRSR-DOWN]winnaar[SPACE]"n$(x
       ):goto22
12    ifts(x)>1thens(x)=ts(x)
13     ifx=1thenprint"[SPACE]"s(x)
14     ifx=2thenprint,"[SPACE]"s(x)
15     ifx=3thenprint,,"[SPACE]"s(x)
16     ifx=4thenprint,,,"[SPACE]"s(x)
17     ifx=1thenprint"[2xCRSR-UP]",
18     ifx=2thenprint"[2xCRSR-UP]",,
19     ifx=3thenprint"[2xCRSR-UP]",,,
20     ifx=4thenprint"[2xCRSR-UP]",,,,
21     next:print"[CRSR-DOWN]":goto10
22    end
23    rem spelregels
24    print"[SHIFT-CLR][10xSPACE]spelreg
      els[SPACE]darts":print
25    print"hang[SPACE]het[SPACE]dart[SP
       ACE]bord[SPACE]op[SPACE]zoals[SPAC
       E]de[SPACE]tekening[SPACE]laat[SPA
       CE]zien,[SPACE]";
26     print"en[SPACE]ga[SPACE]tijdens[SP
       ACE]het[SPACE]spel[SPACE]op[SPACE]
       de[SPACE]juiste[SPACE]afstand[SPAC
       E]staan.":print
27    print"[2xSPACE][COM-L]":forx=1to4:
      print"[2xSPACE][COM-L][COM-K]":nex
      t:forx=1to5:print"[2xSPACE][COM-L]
       ":next
28    print"[2xSPACE][COM-L][23xCOM-P]":
      print"[3xSPACE]<---[SPACE]2,37[S
      PACE]meter[SPACE]--->"
29     print"[HOME][9xCRSR-DOWN]^";:print
       tab(4)"de[SPACE]r00s[SPACE]1,72[SP
       ACE]meter"
30    forx=1to7:print"[SHIFT-]":next :p
      rint
31     print"[2xCRSR-DOWN]druk[SPACE]retu
       rn[SPACE]voor[SPACE]de[SPACE]punte
       n[SPACE]telling"
32     i$="":geti$:ifi$<>chr$(13)then32
33     print"[SHIFT-CLR][10xSPACE]darts[S
       PACE]punten[SPACE]telling":print
34     print"de[SPACE]buitenste[SPACE]dun
       ne[SPACE]ring[SPACE]geeft[SPACE]ee
       n[SPACE]dubbel[SPACE]score[3xSPACE
       ](2x)":print
35     print"de[SPACE]binnenste[SPACE]dun
       ne[SPACE]ring[SPACE]geeft[SPACE]ee
       n[SPACE]triple[SPACE]score[3xSPACE
       ](3x)":print
36     print"de[SPACE]bul[SPACE](roos)[SP
       ACE]geeft[SPACE]een[SPACE]dubbel[S
       PACE]score[4xSPACE]van[SPACE]25[SP
       ACE](dus[SPACE]50)":print
37     print"de[SPACE]ring[SPACE]om[SPACE
       ]de[SPACE]bul[SPACE](roos)[SPACE]g
       eeft[SPACE]een[6xSPACE]enkele[SPAC
       E]score[SPACE]van[SPACE]25":print
38     print"alle[SPACE]overgebleven[SPAC
       E]vakken[SPACE]geven[SPACE]een[6xS
       PACE]enkele[SPACE]score[SPACE](1x)
      "
39     print"[2xCRSR-DOWN]druk[SPACE]retu
       rn[SPACE]voor[SPACE]de[SPACE]spelw
      ijze"
40    i$="":geti$:ifi$<>chr$(13)then40
41     print"[SHIFT-CLR][10xSPACE]darts[S
       PACE]spelwijze":print
42     print"u[SPACE]kiest[SPACE]tussen[S
       PACE]spel[SPACE]301[SPACE]en[SPACE
       ]501":print
43     print"men[SPACE]moet[SPACE]zo[SPAC
       E]snel[SPACE]mogelijk[SPACE]op[SPA
       CE]nul[SPACE]zien[3xSPACE]te[SPACE
       ]komen":print
44     print"hierbij[SPACE]moet[SPACE]de[
       SPACE]laatse[SPACE]worp[SPACE]een[
       SPACE]dubbel[2xSPACE]zijn":print
45     print"per[SPACE]beurt[SPACE]mag[SP
       ACE]u[SPACE]drie[SPACE]darts[SPACE
       ]gooien,"
46     print"om[SPACE]uit[SPACE]te[SPACE]
       gaan[SPACE]op[SPACE]nul[SPACE]ook[
       SPACE]minder":print
47     print"degene[SPACE]die[SPACE]2[SPA
       CE]van[SPACE]de[SPACE]3[SPACE]part
       ijen[SPACE]wint[SPACE]is[2xSPACE]d
       e[SPACE]kampioen"
48     print"[2xCRSR-DOWN]druk[SPACE]retu
       rn[SPACE]voor[SPACE]spel[SPACE]men
      u"
49    i$="":geti$:ifi$<>chr$(13)then49
50     goto4
```

EINDE LISTING darts

Checksum Darts

regel 1	37	regel 29	10
regel 2	59	regel 30	26
regel 3	240	regel 31	205
regel 4	172	regel 32	201
regel 5	213	regel 33	170
regel 6	231	regel 34	222
regel 7	122	regel 35	244
regel 8	1	regel 36	11
regel 9	14	regel 37	196
regel 10	172	regel 38	150
regel 11	215	regel 39	161
regel 12	98	regel 40	200
regel 13	70	regel 41	126
regel 14	115	regel 42	88
regel 15	160	regel 43	201
regel 16	205	regel 44	46
regel 17	152	regel 45	255
regel 18	197	regel 46	36
regel 19	242	regel 47	66
regel 20	31	regel 48	196
regel 21	206	regel 49	209
regel 22	128	regel 50	188
regel 23	133		
regel 24	183		
regel 25	243		
regel 26	147		
regel 27	227		
regel 28	183		

Programma woordtraining

Eindelijk weer eens een educatief programma in deze rubriek. Na het intypen en runnen van dit spel heeft U een taal programma tot uw beschikking. De uitleg is in het spel zelf opgenomen en het geheel is geschreven door G.H. Kemps uit Woerden.

```
100   rem ******************************
      *
120   rem ***                          **
      *
140   rem ***   raamwerk voor het      **
      *
160   rem ***   maken en oefenen van   **
      *
180   rem ***   moeilijke woorden      **
      *
200   rem ***                          **
      *
220   rem ******************************
      *
240   rem ***                          **
      *
250   rem *** door:g.h.kamp            **
      *
260   rem ***                          **
      *
280   rem ***                          **
      *
300   rem ******************************
      *
320   rem ***                          **
      *
340   rem *** gebruik power cartridge **
      *
360   rem ***     is noodzakelijk      **
      *
380   rem ***                          **
      *
400   rem ******************************
      *
420   rem ***                          **
      *
440   rem *** fouten verbeteren in de **
      *
460   rem *** oefening met de _ toets **
      *
480   rem ***                          **
      *
500   rem ******************************
      *
520   :
540   pudef:rem *** runst/rest uit ***
560   :
580   rem *** codenummer ***
600   printchr$(147)chr$(14):dclear8,6,0
       :trap1
620    print"[SPACE]Met[SPACE]dit[SPACE]t
       aalprogramma[SPACE]kunnen"
640    print"[SPACE]moeilijke[SPACE]woord
       en[SPACE]worden[SPACE]geoefend.":t
       rap2:print:print
660    print"[SPACE]Ieder[SPACE]kan[SPACE
       ]zijn/haar[SPACE]eigen[SPACE]gewen
       ste"
680   print"[SPACE]bestand(en)[SPACE]mak
       en.":trap2:print:print
700    print"[SPACE]Om[SPACE]deze[SPACE]b
       estanden[SPACE]tegen[SPACE]verkeer
       d[SPACE]"
720    print"[SPACE]gebruik[SPACE]van[SPA
       CE]anderen[SPACE]te[SPACE]beveilig
      en"
740   print"[SPACE]moet[SPACE]gebruik[SP
       ACE]gemaakt[SPACE]worden[SPACE]van
       [SPACE]een":trap2:print:print
760    print"[2xSPACE][CTRL-9][6xSPACE]o[
       SPACE]n[SPACE]z[SPACE]i[SPACE]c[SP
       ACE]h[SPACE]t[SPACE]b[SPACE]a[SPAC
       E]a[SPACE]r[8xSPACE]":trap1
780    print"[2xSPACE][CTRL-9][6xSPACE]c[
       SPACE]o[SPACE]d[SPACE]e[SPACE]-[SP
       ACE]n[SPACE]u[SPACE]m[SPACE]m[SPAC
       E]e[SPACE]r[3xSPACE][5xSHIFT-SPACE
      ][CTRL-0]":trap2
800    print:print"[SPACE]Typ[SPACE]nu[SP
       ACE]een[SPACE]nummer[SPACE]+[SPACE
       ]RETURN";:printchr$(129);:input co
       de
820   print:printchr$(144)"[SPACE]OK,[SP
       ACE]gebruik[SPACE]dit[SPACE]nummer
       ":trap1
840   print"[SPACE]als[SPACE]daarom[SPAC
       E]wordt[SPACE]gevraagd!":trap2:pri
       nt
860   print:print"[2xSPACE][CTRL-9][SPAC
       E]druk[SPACE]een[SPACE]toets[20xSP
      ACE]"
880   getk$:ifk$="" then 880
900   :
920   rem ***    titelpagina   ***
940    printchr$(147)chr$(142):poke53280,
       8:poke53281,6:print:trap2
960   print"[CTRL-8]*******************
       *******************":trap2
1000   print"[3xCRSR-DOWN]"
1020  printtab(2);"[CTRL-9][CTRL-1][6xSP
      ACE][CTRL-0][5xSPACE][CTRL-9][5xSP
      ACE][CTRL-0][5xSPACE][CTRL-9][5xSP
       ACE][CTRL-0][5xSPACE][CTRL-9][2xSP
      ACE][CTRL-0]"
1040  printtab(4);"[CTRL-9][2xSPACE][CTR
       L 0][7xSPACE][CTRL-9][SPACE][CTRL-
       0][3xSPACE][CTRL-9][SPACE][CTRL-0]
      [5xSPACE][CTRL-9][SPACE][CTRL-0][3
      xSPACE][CTRL-9][SPACE][CTRL-0][5xS
      PACE][CTRL-9][2xSPACE][CTRL-0]"
1060  printtab(4);"[CTRL-9][2xSPACE][CTR
      L 0][7xSPACE][CTRL-9][SPACE][CTRL-
      0][3xSPACE][CTRL-9][SPACE][CTRL-0]
      [5xSPACE][CTRL-9][SPACE][CTRL-0][3
      xSPACE][CTRL-9][SPACE][CTRL-0][5xS
       PACE][CTRL-9][2xSPACE][CTRL-0]"
1080   printtab(4);"[CTRL-9][2xSPACE][CTR
      L 0][7xSPACE][CTRL-9][5xSPACE][CTR
      L 0][5xSPACE][CTRL-9][5xSPACE][CTR
      L 0][5xSPACE][CTRL-9][2xSPACE][CTR
      L 0]"
1100   printtab(4);"[CTRL-9][2xSPACE][CTR
      L 0][7xSPACE][CTRL-9][SPACE][CTRL-
      0][3xSPACE][CTRL-9][SPACE][CTRL-0]
      [5xSPACE][CTRL-9][SPACE][CTRL-0][3
```

```
      xSPACE][CTRL-9][SPACE][CTRL-0][5xS
       PACE][CTRL-9][2xSPACE][CTRL-0]"
1120   printtab(4);"[CTRL-9][2xSPACE][CTR
      L 0][7xSPACE][CTRL-9][SPACE][CTRL-
      0][3xSPACE][CTRL-9][SPACE][CTRL-0]
      [5xSPACE][CTRL-9][SPACE][CTRL-0][3
      xSPACE][CTRL-9][SPACE][CTRL-0][5xS
       PACE][CTRL-9][2xSPACE][CTRL-0]"
1140   printtab(4);"[CTRL-9][2xSPACE][CTR
      L 0][7xSPACE][CTRL-9][SPACE][CTRL-
      0][3xSPACE][CTRL-9][SPACE][CTRL-0]
      [5xSPACE][CTRL-9][SPACE][CTRL-0][3
      xSPACE][CTRL-9][SPACE][CTRL-0][5xS
       PACE][CTRL-9][5xSPACE][CTRL-0]"
1160   b$="m[SPACE]o[SPACE]e[SPACE]i[SPAC
       E]l[SPACE]i[SPACE]j[SPACE]k[SPACE]
       e[3xSPACE]w[SPACE]o[SPACE]o[SPACE]
       r[SPACE]d[SPACE]e[SPACE]n"
1180   print:print:trap2:fora=1tolen(b$):
       printtab(4)mid$(b$,a,1);
1200   forp=1to10:nextp:nexta
1220   print"[4xCRSR-DOWN]":trap2
1240  print"[CTRL-8]*******************
       ********************":trap2
1280  :
1300  rem *** schoonmaken scherm ***
1320   va=1024:ka=55296
1340  for rn=0to 24:for kn=0to 39
1360  poke va+kn+40*rn,160
1380  poke ka+kn+40*rn,8
1400  next kn:next rn:trap1
1420  :
1440  rem *** informatie gebruik ***
1460  poke53280,8:poke53281,8:printchr$(
       144)
1480  printchr$(14):printchr$(18):print"
       [HOME]"
1500   print:print"======================
       ==================="
1520  print"[3xCRSR-DOWN]";
1540  print"In[SPACE]dit[SPACE]programma
       [SPACE]zijn[SPACE]4[SPACE]blz.[SPA
       CE]informatie[SPACE]over[SPACE]het
       [SPACE]gebruik[SPACE]opgenomen.":
1560   print"Je[SPACE]kan[SPACE]nu[SPACE]
       kiezen[SPACE]of[SPACE]je[SPACE]daa
       rvan[SPACE]gebruik[2xSPACE]wilt[SP
       ACE]maken.":print:print:print
1580   print"============================
       ============"
1600  print"[4xCRSR-DOWN]"
1620   print"[CTRL-9][SPACE]maak[SPACE]ee
       n[SPACE]keuze[SPACE].............
      .....([CTRL-0]j[CTRL-9]/[CTRL-0]n[
      CTRL-9])"
1640  get k$:if k$="" then 1640
1660  if k$<>"j" and k$<>"n" then 1640
1680  if k$="n" thenprintchr$(147):goto3
      040
1700   :
1720  rem *** informatie blz 1 ***
1740   printchr$(147)
1760  printchr$(18):print"[HOME]"
1780  printtab(7);"[CTRL-9][SPACE]inform
       atie[SPACE]taalprogramma[SPACE][CT
      RL 0]":print:print
1800   print"============================
       ============":trap1
1820   print"Dit[SPACE]programma[SPACE]is
       [SPACE]'menu-gestuurd'.":trap2
1840   print:print"Om[SPACE]te[SPACE]voor
       komen[SPACE]dat[SPACE]bij[SPACE]on
       juist[SPACE]gebruik":trap1
1860   print"het[SPACE]woordenbestand[SPA
       CE]verloren[SPACE]gaat[SPACE]wordt
       ":trap1
1880  print"steeds[SPACE]om[SPACE]het[SP
       ACE]invoeren[SPACE]van[SPACE]het[S
       PACE]code-":trap1
1900   print"nummer[SPACE]gevraagd.":trap
      2:print
1920  print"Dit[SPACE]codenummer[SPACE]i
       s[SPACE]tijdens[SPACE]het[SPACE]in
       tikken":trap1
1940   print"niet[SPACE]zichtbaar.":trap2
      :print
1960  print"Bij[SPACE]een[SPACE]onjuist[
       SPACE]nummer[SPACE]kan[SPACE]het[S
       PACE]pro-":trap1
1980   print"gramma[SPACE]niet[SPACE]word
       en[SPACE]voortgezet[SPACE]en":trap
       1
2000  print"keert[SPACE]naar[SPACE]het[S
       PACE]menu[SPACE]terug.":print
2020   print"============================
       ============":trap1
2040  print"[CTRL-9][SPACE]druk[SPACE]op
       [SPACE]een[SPACE]toets[SPACE].....
       ...........>>.2."
2060  get k$:if k$="" then 2060
2080  :
2100  printchr$(147)
2120  rem *** informatie blz 2 ***
2140   print"vervolg[SPACE]informatie[SPA
       CE].........."
2160   print"============================
       ============":trap2
2180   print"Het[SPACE]menu[SPACE]is[SPAC
       E]als[SPACE]volgt[SPACE]samengeste
       ld:"
2200  print"[34xCOM-T]":print:trap2
2220   printtab(3);"invoer[3xSPACE](van[S
       PACE]nieuwe[SPACE]woorden)":trap1
2240   printtab(3);"opslaan[2xSPACE](bewa
       ren[SPACE]op[SPACE]schijf)":trap1
2260   printtab(3);"laden[4xSPACE](van[SP
       ACE]de[SPACE]schijf[SPACE]halen)":
      trap1
2280  printtab(3);"lezen[4xSPACE](contro
       le[SPACE]op[SPACE]fouten)":trap1
2300   printtab(3);"wijzigen[SPACE](verbe
       teren[SPACE]van[SPACE]fouten)":pri
       nt:trap1
2320   printtab(3);"start[4xSPACE](starte
       n[SPACE]van[SPACE]programma)":trap
      1
2340  printtab(3);"stoppen[2xSPACE](stop
       pen[SPACE]van[SPACE]programma)":pr
       int:trap2
2360   print"Wordt[SPACE]gekozen[SPACE]vo
       or[SPACE]'lezen','wijzigen'[SPACE]
       of'start',":trap1
2380   print"dan[SPACE]wordt[SPACE]gevraa
       gd[SPACE]welke[SPACE]woorden[SPACE
       ]je[SPACE]wilt"
2400   print"============================
```

```
      ============":trap1
2420 print"[CTRL-9][SPACE]druk[SPACE]op
      [SPACE]een[SPACE]toets[SPACE].....
      ...........>>.3."
2440 get k$:if k$="" then 2440
2460 :
2480 rem *** informatie blz 3 ***
2500 printchr$(147)
2520  print"vervolg[SPACE]informatie[SPA
      CE]..........":print
2540  print"============================
      ============":print:trap2
2560  print"Je[SPACE]moet[SPACE]dan[SPAC
      E]altijd[SPACE]2[SPACE]getallen[SP
      ACE]opgeven,[2xSPACE]gescheiden[SP
      ACE]door[SPACE]een[SPACE]komma."
2580  trap2:print:print:print"[CTRL-9][S
      PACE]**[SPACE]voorbeeld[SPACE]**[S
     PACE][CTRL-0]":print:trap2
2600  print"25,50[2xSPACE]betekent[SPACE
      ]dat[SPACE]je[SPACE]de[SPACE]woord
      en[SPACE]leest,wijzigt[SPACE]of[SP
      ACE]oefent[SPACE]van[SPACE]25-50[S
      PACE]enz."
2620  trap2:print"Het[SPACE]zetten[SPACE
      ]van[SPACE]een[SPACE]punt[SPACE]ge
      eft[SPACE]een[SPACE]error.":print
2640  print"============================
      ============":print:trap1
2660 print"[CTRL-9][SPACE]druk[SPACE]op
      [SPACE]een[SPACE]toets[SPACE].....
      ...........>>.4."
2680 :
2700 get k$:if k$="" then 2700
2720 rem *** informatie blz 4 ***
2740 printchr$(147)
2760  print"vervolg[SPACE]informatie[SPA
      CE]..........":print
2780  print"============================
      ============":print:trap2
2800  print"Nog[SPACE]enkele[SPACE]opmer
      kingen:":print:print:trap2
2820  print"Het[SPACE]bestand[SPACE]kan[
      SPACE]in[SPACE]dit[SPACE]programma
      [SPACE]max.[3xSPACE]500[SPACE]woor
      den[SPACE]groot[SPACE]zijn."
2840 trap2
2860  print:print"Het[SPACE]is[SPACE]ver
      standiger[SPACE]om[SPACE]kleinere[
      SPACE]bestan-[SPACE]den[SPACE]aan[
      SPACE]te[SPACE]leggen."
2880 trap2
2900  print:print"Geef[SPACE]in[SPACE]da
      t[SPACE]geval[SPACE]ieder[SPACE]be
      stand[SPACE]een[5xSPACE]andere[SPA
      CE]naam.":print
2920  print"============================
      ============":print:trap1
2940 print"[CTRL-9][SPACE]nog[SPACE]een
      [SPACE]keer.?[9xSPACE]j/n[5xSPACE]
      <<[SPACE]>>[3xSPACE]"
2960 get k$:if k$="" then 2960
2980 if k$<>"j" and k$<>"n" then2960
3000 if k$="j" then 1700
3020  :
3040 gosub10020:x=500:dim w$(x):printch
      r$(14)chr$(8)
3060 x$="[CTRL-9][2xSPACE]**[SPACE]moei
      jlijke[SPACE]woorden[SPACE]oefenen
      [SPACE]**[2xSPACE]"
3080  printchr$(147);:dclear5,5,0:printt
      ab(2)x$:print:print
3100 printtab(2)"[CTRL-9][SPACE]f2[SPAC
     E][CTRL-0][SPACE]invoer[2xSPACE]":
     print
3120  printtab(2)"[CTRL-9][SPACE]f4[SPAC
     E][CTRL-0][SPACE]opslaan[SPACE]":p
     rint
3140  printtab(2)"[CTRL-9][SPACE]f6[SPAC
     E][CTRL-0][SPACE]laden[3xSPACE]":p
     rint
3160  printtab(2)"[CTRL-9][SPACE]f8[SPAC
     E][CTRL-0][SPACE]lezen[3xSPACE]":p
     rint
3180  printtab(2)"[CTRL-9][SPACE]f1[SPAC
     E][CTRL-0][SPACE]wijzigen":print:p
     rint
3200 for s=1 to 36:printtab(2)"[COM-U]"
     ;:next s:print:print
3220 print:printtab(2)"[CTRL-9][SPACE]f
     7[SPACE][CTRL-0][SPACE]start[3xSPA
     CE]"
3240  print:printtab(2)"[CTRL-9][SPACE]f
     5[SPACE][CTRL-0][SPACE]stoppen[SPA
     CE]"
3260 get k$:if k$="" then 3260
3280 if k$="[F2]" then gosub 3460
3300 if k$="[F4]" then gosub 4120
3320 if k$="[F6]" then gosub 4640
3340 if k$="[F8]" then gosub 5300
3360 if k$="[F1]" then gosub 5680
3380 if k$="[F7]" then gosub 6560
3400 if k$="[F5]" then 10200
3420 goto 3060
3440  :
3460 rem ** invoer **
3480  print"[SHIFT-CLR]";:printtab(2);x$
      :print:print:print:trap 1
3500  printtab(2);"Aantal[SPACE]geladen[
      SPACE]woorden[3xSPACE]";"[CTRL-9]"
      ;w-1:print:print:trap 3
3520  printtab(2);"Als[SPACE]er[SPACE]wo
      orden[SPACE]aan[SPACE]het[SPACE]be
      stand[SPACE]"
3540  printtab(2);"moeten[SPACE]worden[S
      PACE]toegevoegd,[SPACE]moet[SPACE]
      het[SPACE]"
3560  printtab(2);"eerst[SPACE]worden[SP
      ACE]geladen[SPACE].............":p
      rint:print:print:trap 2
3580 printtab(2)"[CTRL-9]Zijn[SPACE]de[
      SPACE]woorden[SPACE]geladen?[4xSPA
     CE][CTRL-0][SPACE]j/n[SPACE][CTRL-
     9][4xSPACE][CTRL-0]":trap 2
3600 printtab(2)"[39xSPACE]"
3620  printtab(2)"[CTRL-9][37xSPACE][CTR
     L 0]"
3640 printtab(2)"[CTRL-9][37xSPACE][CTR
     L 0]"
3660 printtab(2)"[CTRL-9][37xSPACE][CTR
      L 0]"
3680 printtab(2)"[CTRL-9][37xSPACE][CTR
     L 0]"
3700  printtab(2)"[CTRL-9][37xSPACE][CTR
     L 0]":trap 2
3720 print"[5xCRSR-UP]"
```

```
3740  printtab(2)"[CTRL-9]Geen[SPACE]woo
      rden[SPACE]geladen[SPACE]mag[SPACE
      ]alleen[SPACE]bij[2xSPACE][CTRL-0]
      ":trap1
3760  printtab(2)"[CTRL-9]het[SPACE]make
      n[SPACE]van[SPACE]een[SPACE]nieuw[
      SPACE]bestand!![4xSPACE][CTRL-0]":
      trap1
3780  printtab(2)"[CTRL-9]Er[SPACE]moet[
      SPACE]dan[SPACE]ook[SPACE][CTRL-0]
      [SPACE]j[SPACE][CTRL-9][SPACE]word
      en[SPACE]gekozen![SPACE]":trap1
3800  get k$:if k$="" then 3800
3820  if k$<>"j" and k$<>"n" then 3800
3840  if k$="n" then return
3860  if k$="j" then print"[3xCRSR-DOWN]
      ";tab(2)"Geef[SPACE]het[SPACE][CTR
      L 9][SPACE]codenummer[SPACE][CTRL-
      0]";chr$(30):input c
3880  if c<>code then return:printchr$(1
      44)
3900  printchr$(144);"[SHIFT-CLR]";:prin
      ttab(2);x$:print
3920  if w>=x-1 then return
3940  printtab(2);"[CTRL-9][SPACE]invoer
      [SPACE]woorden[SPACE]"
3960  print
3980  w=w+1
4000  printtab(2);"Aantal[SPACE]woorden:
      ";w;"van[SPACE]de[SPACE]";x:print
4020  printtab(2);"Laatste[SPACE]woord[S
      PACE]->>[SPACE]Toets[2xSPACE][CTR
      L 9][SPACE]0[SPACE][CTRL-0]"
4040  print"[5xCRSR-DOWN][7xSPACE]";:inp
      ut "woord";w$(w)
4060  if w$(w)="0"then print"[SHIFT-CLR]
      ":return
4080  goto 3900
4100  :
4120  rem ** opslaan **
4140  print"[SHIFT-CLR]";:printtab(2);x$
      :print:print:trap 1
4160  printtab(2);"[CTRL-9][SPACE]WAARSC
      HUWING[SHIFT-SPACE][CTRL-0].......
      ...........":print:print:trap2
4180  printtab(2);"Als[SPACE]aan[SPACE]e
      en[SPACE]bestand[SPACE]woorden[SPA
      CE]moeten[SPACE]":trap1
4200  printtab(2);"worden[SPACE]toegevoe
      gd,[SPACE]wordt[SPACE]dat[SPACE]ge
      wist,":trap1
4220  printtab(2);"indien[SPACE]het[SPAC
      E]niet[SPACE]eerst[SPACE]is[SPACE]
      geladen!":print:print:print:trap2
4240  printtab(2);"Aantal[SPACE]geladen[
      SPACE]woorden[3xSPACE]";"[CTRL-9]"
      ;w-1:print:print:print:trap 2
4260  printtab(2);"Indien[SPACE]u[SPACE]
      wilt[SPACE]opslaan[SPACE]moet[SPAC
      E]u[SPACE]eerst":trap1
4280  printtab(2);"uw[SPACE]codenummer[S
      PACE]geven!":print:print:trap 2
4300  printtab(2);"[CTRL-9][SPACE]codenu
      mmer[SPACE][CTRL-0]";chr$(30):inpu
      t c
4320  if c<>code then printchr$(144):ret
      urn
4340  printchr$(144);"[SHIFT-CLR]";:prin
      ttab(2);x$:print
4360  printtab(2)"geef[SPACE]bestandsnaa
      m[SPACE]->[SPACE]";:gosub 9780
4380  fl$="@0:"+f$+",seq,write"
4400  printchr$(144);"[SHIFT-CLR]";:prin
      ttab(2);x$:print
4420  printtab(2);"[CTRL-9][SPACE]"f$"[S
      PACE][CTRL-0][SPACE]wordt[SPACE]op
      geslagen[SPACE]!"
4440  print:printtab(2);"even[SPACE]gedu
      ld[SPACE]a.u.b.[2xSPACE]->[SPACE]-
      >[SPACE]->";:print"[2xCRSR-UP]"
4460  open1,8,3,fl$
4480  print#1,w
4500  for i=1 to w
4520  print:printtab(34);"[CTRL-9]";i;"[
      2xCRSR-UP]"
4540  print#1,w$(i)
4560  next i
4580  close 1
4600  print"[SHIFT-CLR]":fori=1tow:w$(i)
      ="":nexti:w=0:goto3060
4620  :
4640  rem ** laden **
4660  fori=1tow:w$(i)="":nexti:w=0
4680  print"[SHIFT-CLR]";:printtab(2);x$
      :print
4700  printtab(2)"Geef[SPACE]bestandsnaa
      m[SPACE]->[SPACE]";:gosub9780:prin
      t
4720  print:printtab(2);"zoeken[SPACE]na
      ar:[SPACE][CTRL-9][SPACE]"f$:trap1
4740  print:printtab(2);"even[SPACE]gedu
      ld[SPACE]a.u.b.[2xSPACE]->[SPACE]-
      >[SPACE]->";:print"[2xCRSR-UP]"
4760  open1,8,3,"0:"+f$+",seq,read"
4780  input#1,w:z=w
4800  ifz=0thenprinttab(7)"[10xCRSR-DOWN
      ][CTRL-9][SPACE]Bestand[SPACE]niet
      [SPACE]aanwezig![SPACE]":trap2:got
      o5220
4820  print:printtab(2);"even[SPACE]gedu
      ld[SPACE]a.u.b.[2xSPACE]->[SPACE]-
      >[SPACE]->";:print"[2xCRSR-UP]"
4840  for i=1 to w
4860  print:printtab(34);"[CTRL-9]";i;"[
      2xCRSR-UP]"
4880  input#1,w$(i)
4900  next i
4920  close 1
4940  :
4960  rem ** teken einde laden **
4980  mu=54272
5000  for k=1 to 3
5020  for s=0to 24:pokemu+s,0:nexts
5040  fort1=1 to 5
5060  pokemu+24,15:pokemu+5,9:pokemu+4,1
      7
5080  fortt=1to20:next
5100  pokemu+1,70:pokemu,50
5120  fortt=1to20:next
5140  pokemu+1,90:pokemu,50:pokemu+4,0
5160  nextt1:pokemu+1,0:pokemu,0
5180  nextk:trap1
5200  print"[SHIFT-CLR]":return
5220  close 1:err$"ui
5240  print"[SHIFT-CLR]";:printtab(2)"[C
      TRL 9][SPACE]Inhoud[SPACE]van[SPAC
```

```
      E]de[SPACE]schijf:[2xSPACE]":trap2
5260  catalog:trap2:printtab(2)"[CTRL-9]
      [SPACE]Zoek[SPACE]het[SPACE]goede[
      SPACE]bestand[SPACE]":trap10:return
5280  :
5300  rem ** lezen **
5320  if w=0then return
5340  print"[SHIFT-CLR]";:printtab(2);x$
       :print:print:print:print:print:pri
      nt:print:trap 1
5360  printtab(2);"Geef[SPACE]eerst[SPAC
       E]uw[SPACE]codenummer":print:print
      :trap 1
5380  printtab(2);"[CTRL-9][SPACE]codenu
       mmer[SPACE][CTRL-0]";chr$(30):inpu
      t c
5400  if c<>code then printchr$(144):ret
       urn
5420  printchr$(144);"[SHIFT-CLR]":print
       tab(2);"[CTRL-9][SPACE]lezen[SPACE
       ]van[SPACE]de[SPACE]woorden[SPACE]
       ":print
5440  printtab(2);"Aantal[SPACE]woorden:
       ";w-1
5460  print"[2xSPACE]";:input "Welke[SPA
       CE]woorden[SPACE]wil[SPACE]je[SPAC
       E]lezen";a,b:print
5480  for i=a to b
5500   printtab(4);i;:printtab(15);w$(i)
5520  if i/20<>int(i/20)then 5600
5540  get k$:if k$="" then 5540
5560  printchr$(144);"[SHIFT-CLR]":print
      tab(2);"[CTRL-9][2xSPACE]lezen[SPA
       CE]van[SPACE]de[SPACE]woorden[SPAC
      E]:[CTRL-0]";
5580  printtab(25);a;"-";b:print
5600  next i
5620  get k$:if k$="" then 5620
5640   print"[SHIFT-CLR]":return
5660  :
5680  rem ** wijzigen **
5700  print"[SHIFT-CLR]";:printtab(2);x$
      :print:print:print:print:trap 1
5720  printtab(2);"Aantal[SPACE]geladen[
       SPACE]woorden[2xSPACE]";"[CTRL-9]"
       ;w-1:print:print:print:print
5740  trap2
5760   printtab(2);"Geef[SPACE]eerst[SPAC
       E]uw[SPACE]codenummer":print:print
      :trap 1
5780  printtab(2);"[CTRL-9][SPACE]codenu
       mmer[SPACE][CTRL-0]";chr$(30):inpu
      t c
5800  if c<>code then printchr$(144):ret
       urn
5820  printchr$(144);"[SHIFT-CLR]";:prin
       ttab(2);x$:print
5840  printtab(2);"[CTRL-9][SPACE]wijzig
       en[SPACE]van[SPACE]de[SPACE]woorde
       n[SPACE]":print
5860  print:printtab(2);"[CTRL-9][SPACE]
      f1[SPACE][CTRL-0][SPACE]alles[SPAC
      E]doorbladeren"
5880   print:printtab(2);"[CTRL-9][SPACE]
      f3[SPACE][CTRL-0][SPACE]zoeken":pr
      int
5900  get k$:if k$="" then 5900
5920  if k$="[F1]" then 5960
5930  if k$="[F3]" then 6260
5940  goto 5900
5960  print"[SHIFT-CLR]";:printtab(2);x$
       :print"[2xCRSR-DOWN]"
5980   printtab(5);"Totaal[SPACE]aantal[S
       PACE]woorden:"w-1:print:print:print
6000  print"[5xSPACE]";:input "Welke[SPA
       CE]woorden[SPACE]wil[SPACE]je";a,b
6020  for i=a to b
6040  print"[SHIFT-CLR]";:printtab(2);x$
       :print"[2xCRSR-DOWN]"
6060   printtab(5);"Totaal[SPACE]aantal[S
       PACE]woorden:"w-1:print:print:print
6080  printtab(5);i;:printtab(15);w$(i)
6100   print:print:print:print:printtab(5
      );"[CTRL-9][SPACE]f1[SPACE][CTRL-0
       ][SPACE]verder[SPACE]gaan"
6120   print:printtab(5);"[CTRL-9][SPACE]
      f3[SPACE][CTRL-0][SPACE]wijzigen":
      print
6140  get k$:if k$="" then 6140
6160  if k$="[F1]" then 6220
6180  if k$="[F3]" then gosub 6480:goto
       6220
6200  goto 6140
6220  next i
6240  print"[SHIFT-CLR]":return
6260  print"[SHIFT-CLR]";:printtab(2);x$
      :print"[6xCRSR-DOWN]"
6280  print"[2xSPACE]";:input "geef[SPAC
       E]zoekwoord";z$
6300  for i=1 to w
6320  if w$(i)=z$ then gosub 6440:return
6340  next i
6360  print:print:print:print:printtab(8
       );"[CTRL-9][SPACE]woord[SPACE]niet
       [SPACE]gevonden[2xSPACE]"
6380  printtab(8);"[CTRL-9][SPACE]druk[S
       PACE]toets[SPACE]voor[SPACE]menu[S
       PACE]"
6400  get k$:if k$="" then 6400
6420   print"[SHIFT-CLR]":return
6440  print"[SHIFT-CLR]";:printtab(2);x$
      :print"[5xCRSR-DOWN]"
6460  printtab(5);w$(i):print
6480   printtab(5);"wijzigen[SPACE]van[SP
       ACE]het[SPACE]woord":print:print
6500   print"[5xSPACE]";:input "gewijzigd
       [SPACE]woord:";d$:w$(i)=d$:i=i-1
6520  print"[SHIFT-CLR]":return
6540  :
6560  rem ** start programma **
6580  print"[SHIFT-CLR]";:printtab(2);x$
      :print"[2xCRSR-DOWN]"
6600  print"[6xCRSR-DOWN]";:printtab(3);
       "We[SPACE]gaan[SPACE]nu[SPACE]same
       n[SPACE]woorden[SPACE]oefenen."
6620   print:print:printtab(3);:input"Hoe
       [SPACE]heet[SPACE]je[2xSPACE]";n$
6640  trap1
6660  print"[SHIFT-CLR]";:printtab(2);x$
       :print:print
6680   printtab(3)"Hallo,[SPACE]";n$:trap1
6700  print:printtab(3)"Je[SPACE]kan[SPA
       CE]straks[SPACE]kiezen:":trap2:pri
       nt"[CRSR-DOWN]"
6720  printtab(3)"[CTRL-9][SPACE]1[SPACE
      ][CTRL-0][SPACE]Hoe[SPACE]lang[SPA
```

```
      CE]je[SPACE]elk[SPACE]woord[SPACE]
      wilt[SPACE]zien.":trap1
6740  printtab(3)"[CTRL-9][SPACE]2[SPACE
      ][CTRL-0][SPACE]Welke[SPACE]woorde
      n[SPACE]je[SPACE]wilt[SPACE]oefene
      n.":trap2:print:print
6760  printtab(3)"Je[SPACE]ziet[SPACE]st
      eeds[SPACE]een[SPACE]woord"
6780  printtab(3)"dat[SPACE]je[SPACE]moe
      t[SPACE]natypen.":trap2:print
6800  printtab(3)"Maak[SPACE]je[SPACE]ee
      n[SPACE]fout,[SPACE]dan[SPACE]zie[
      SPACE]je[SPACE]dat"
6820  printtab(3)"woord[SPACE]weer,[SPAC
      E]maar[SPACE]iets[SPACE]langer.":t
      rap2:print
6840  printtab(3)"Maak[SPACE]je[SPACE]ge
      en[SPACE]fout,[SPACE]dan[SPACE]kom
      t[SPACE]er[SPACE]weer"
6860  printtab(3)"een[SPACE]ander[SPACE]
      woord!!":trap2
6880  print"[2xCRSR-DOWN][2xSPACE][CTRL-
      9][5xSPACE]druk[SPACE]op[SPACE]een
      [SPACE]toets[2xSPACE]...........[
      CTRL-0]"
6900  getk$:ifk$="" then 6900
6920   :
6940  dclear 5,5,0
6960  rem ** instellen tijd per woord **
6980  print"[SHIFT-CLR]";:printtab(2);x$
      :print"[2xCRSR-DOWN]"
7000  printtab(2);"Hoe[SPACE]lang[SPACE]
      wil[SPACE]je[SPACE]elk[SPACE]woord
      [SPACE]zien[SPACE]"
7020  printtab(2);"[3xCOM-T][SHIFT-SPACE
      ][4xCOM-T][SHIFT-SPACE][3xCOM-T][S
      HIFT SPACE][2xCOM-T][SHIFT-SPACE][
      3xCOM-T][SHIFT-SPACE][5xCOM-T][SHI
      FT SPACE][4xCOM-T]":print:print
7040  printtab(2);"[CTRL-9][SPACE]1[SPAC
      E][CTRL-0]:[SPACE]een[SPACE]second
      e":print
7060  printtab(2);"[CTRL-9][SPACE]2[SPAC
      E][CTRL-0]:[SPACE]twee[SPACE]secon
      den":print
7080  printtab(2);"[CTRL-9][SPACE]3[SPAC
      E][CTRL-0]:[SPACE]drie[SPACE]secon
      den":print
7100  printtab(2);"[CTRL-9][SPACE]4[SPAC
      E][CTRL-0]:[SPACE]vier[SPACE]secon
      den":print
7120  printtab(2);"[CTRL-9][SPACE]5[SPAC
      E][CTRL-0]:[SPACE]vijf[SPACE]secon
      den":print:print:print
7140  printtab(2);:input "maak[SPACE]een
      [SPACE]keus";t
7160  if t<1 or t>5 then 6980
7180  printchr$(147);:printtab(2);x$:print
7200  printtab(2);"Totaal[SPACE]aantal[S
      PACE]woorden:"w-1:print:print
7220  print"[2xSPACE]";:input "Welke[SPA
      CE]woorden[SPACE]wil[SPACE]je[SPAC
      E]oefenen";a,b
7240  print"[9xCRSR-DOWN]":printtab(2)"D
      enk[SPACE]erom.......":trap1
7260  print:printtab(2)"Fouten[SPACE]ver
      beter[SPACE]je[SPACE]met[SPACE]de[
      SPACE][CTRL-9][SPACE]_[SPACE][CTRL
      0][SPACE]toets!":print:print
7280  trap1:print"[2xSPACE][CTRL-9][SPAC
      E]druk[SPACE]een[SPACE]toets[20xSP
      ACE]"
7300  getk$:ifk$="" then 7300
7320  print"[SHIFT-CLR]"
7340  for i=a to b+1
7360   aw=aw+1
7380  if i=b+1 then 8400
7400   print"[HOME]":printtab(2);x$:print
7420   printtab(2);"woord-:";"[CTRL-9]";a
      w;"[CTRL-0]";tab(15);"fout-:";"[CT
      RL 9]";fw;
7440   gosub9340:printleft$(ne$,14);i
7460   gosub9340:printspc(8)
7480   fora=1tolen(w$(i)):printmid$(w$(i)
      ,a,1);:forp=1to10:nextp:nexta
7500   trapt:gosub9340:gosub9520
7520  goto 7580
7540  next i
7560  :
7580  rem ** foutencontrole **
7600   print"[HOME]":printtab(2);x$:print
7620   printtab(2);"woord-:";"[CTRL-9]";a
      w;"[CTRL-0]";tab(15);"fout-:";"[CT
      RL 9]";fw;
7640   if an$<>w$(i) then fw=fw+1:goto792
      0
7660  ifan$=w$(i)thenh=0:gosub9340:print
       "Goed[SPACE]zo,[SPACE]";n$;"!!"
7680  :
7700  rem *** piep ***
7720  mu=54272
7740   for s=0to 24
7760  pokemu+s,0:nexts
7780  pokemu+24,15
7800  fort1=1to5
7820   pokemu+5,9:pokemu+4,17
7840  fortt=1 to 50:next
7860  pokemu+1,70:pokemu,50
7880   nexttl:pokemu+1,0:pokemu,0
7900   trap1:goto7540
7920   print"[HOME]":printtab(2);x$:print
7940   printtab(2);"woord-:";"[CTRL-9]";a
      w;"[CTRL-0]";tab(15);"fout-:";"[CT
      RL 9]";fw
7960   gosub9340
7980   print"[3xCRSR-LEFT]Jammer,[SPACE]"
      ;n$;"[SPACE]kijk[SPACE]nog[SPACE]e
      ens!"
8000  :
8020  rem ** pieper **
8040   mu=54272
8060  for s=0to 24
8080   pokemu+s,0:nexts
8100  pokemu+24,15
8120  for t1=1 to 5
8140  pokemu+5,9:pokemu+4,17
8160   fortt=1 to 20:next
8180  pokemu+1,70:pokemu,50
8200  fortt=1 to 20:next
8220   pokemu+1,90:pokemu,50:pokemu+4,0
8240   nexttl:pokemu+1,0:pokemu,0
8260  :
8280  rem ** herhaling woord **
8300  h=h+1
8320   gosub9340:printleft$(ne$,14);i
8340   gosub9340:printspc(8)
```

```
8360   fora=1tolen(w$(i)):printmid$(w$(i)
       ,a,1);:forp=1to50*h:nextp:nexta
8380   trapt+h:gosub9340:gosub9520:goto76
      00
8400  forx=1 to 3
8420   dclear8,8,0:printchr$(147)tab(2);x
      $:print
8440  :
8460  rem ** afbeelden resultaat **
8480  for a=1to3
8500  gosub 9340
8520  print"[3xSPACE]Je[SPACE]oefende";a
       w-1;"woorden[2xSPACE]!!"
8540  trap 1:gosub 9340
8560  print"[3xSPACE]Je[SPACE]maakte[SPA
       CE]";fw;"fout(en)[2xSPACE]!!"
8580  trap 1
8600  next a:trap2
8620  :
8640  rem ** berekenen resultaat **
8660   e1$="**[SPACE]einde[SPACE]van[SPAC
       E]deze[SPACE]oefening[SPACE]**":e2
       $="wacht[SPACE]even[SPACE]op[SPACE
       ]je[SPACE]beoordeling[SPACE]!"
8680  gosub9340:for u=1to3
8700   fora=1tolen(e1$):printmid$(e1$,a,1
       );:forp=1to3:nextp:nexta:trap 1
8720  gosub 9340
8740  fora=1tolen(e2$):printmid$(e2$,a,1
       );:forp=1to3:nextp:nexta:trap 1
8760  gosub 9340
8780  nextu:trap1
8800  if fw=0then printtab(3);"beoordel
      ing:[SPACE][CTRL-9][SPACE]prima[SP
      ACE]":goto8900
8820  if fw<=aw/10then printtab(3);"beo
       ordeling:[SPACE][CTRL-9][SPACE]goe
       d[SPACE]"
8840  if fw>aw/10and fw<=aw/5 thenprint
      tab(3);"beoordeling:[SPACE][CTRL-9
      ][SPACE]voldoende[SPACE]"
8860  if fw>aw/5 and fw<=aw/4 then print
       tab(3);"beoordeling:[SPACE][CTRL-9
       ][SPACE]zwak[SPACE]"
8880  if fw>aw/4 then printtab(3);"beoor
      deling:[SPACE][CTRL-9][SPACE]onvol
      doende[SPACE]"
8900  trap 1
8920  printleft$(ne$,19);k1$;k2$;k3$;
8940  fork=1to3:printmid$(k2$,2,35)
8960   nextk:printleft$(ne$,20)tab(45);
8980   trap1:print"[3xCRSR-LEFT][CTRL-9][
       SPACE]Nog[SPACE]een[SPACE]keer[SPA
       CE]??[10xSPACE]->[SPACE]->[SPACE]j
       /n[SPACE]"
9000  get k$:if k$="" then 9000
9020  if k$<>"j" and k$<>"n" then 9000
9040   if k$="n"then 9080
9060  if k$="j" then aw=0:fw=0:goto 6940
9080  printchr$(147)"[9xCRSR-DOWN][2xCRS
       R-RIGHT][2xSPACE]Bedankt[SPACE]";n
       $;",":trap1
9100   print:printtab(4)"en[SPACE]tot[SPA
       CE]ziens[SPACE]maar[SPACE]weer[SPA
       CE]!!":trap2
9120  print"[SHIFT-CLR][5xCRSR-DOWN]";ta
       b(4);"Wie[SPACE]wil[SPACE]er[SPACE
       ]nu[SPACE]oefenen[SPACE]?":trap 3
9140   print:printtab(4);"Kom[SPACE]dan[S
       PACE]maar[SPACE]even[SPACE]hier[SP
       ACE]zitten!":trap 5:print:print
9160   print:printtab(4)"Wil[SPACE]je[SPA
       CE]ook[SPACE]....":print
9180  printtab(4)"[CTRL-9][SPACE]"f$"[SP
      ACE][CTRL-0][SPACE]oefenen[SPACE]-
      >[SPACE]j/n[SPACE]":print:print:pr
      int
9200  get k$:if k$="" then 9200
9220  if k$<>"j" and k$<>"n" then 9200
9240   ifk$="j"then 9300
9260   ifk$="n"thenprinttab(4);"[CTRL-9][
       SPACE]Vraag[SPACE]even[SPACE]hulp[
       SPACE]!![SPACE]":trap5
9280   aw=0:fw=0:goto3080
9300  aw=0:fw=0:dclear5,5,0:goto 6560
9320  :
9340  rem *** kaders ***
9360  printchr$(146)
9380   printleft$(ne$,15);
9400  printk1$;k2$;k3$;
9420  fork=1to3:printmid$(k2$,2,35)
9440   nextk:printleft$(ne$,16)tab(45);
9460  return
9480  :
9500  rem *** controle invoer ***
9520  an$="":printspc(8)bs$;
9540  geta$:ifa$="" then 9540
9560  if asc(a$)=13 then return
9580   if asc(a$)=20then 9540
9600  if asc(a$)=95 then 9660
9620  an$=an$+a$
9640   printa$;bs$;:goto 9540
9660  if len(an$)<1 then 9540
9680  an$=left$(an$,len(an$)-1)
9700   printchr$(32)chr$(157)chr$(157)bs$;
9720  goto 9540
9740  :
9760  rem *** controle laden ***
9780  f$="":printbs$;
9800   geta$:ifa$="" then 9800
9820  if asc(a$)=13 then return
9840  if asc(a$)=20then 9800
9860   if asc(a$)=95 then 9920
9880  f$=f$+a$
9900   printa$;bs$;:goto 9800
9920  if len(f$)<1 then 9800
9940  f$=left$(f$,len(f$)-1)
9960   printchr$(32)chr$(157)chr$(157)bs$;
9980  goto 9800
10000 :
10020 rem *** initialiseren ***
10040  ne$=chr$(19):fork=1to20
10060  ne$=ne$+chr$(17):nextk
10080  fork=1to38:kk$=kk$+chr$(96)
10100  l$=l$+chr$(32):nextk
10120  k1$=chr$(176)+kk$+chr$(174)
10140  k2$=chr$(125)+l$+chr$(125)
10160  k3$=chr$(173)+kk$+chr$(189)
10180  bs$=chr$(164)+chr$(157):return
10200 :
10220 rem *** einde programma ***
10240  printchr$(147)chr$(142):dclear6,14
       ,14:pudef:new
```

EINDE LISTING woordtraining

Checksum Woordtraining

regel 100	165
regel 120	139
regel 140	12
regel 160	111
regel 180	66
regel 200	139
regel 220	165
regel 240	139
regel 250	13
regel 260	148
regel 270	159
regel 280	139
regel 300	165
regel 320	139
regel 340	182
regel 360	172
regel 380	139
regel 400	165
regel 420	139
regel 440	110
regel 460	51
regel 480	139
regel 500	165
regel 520	58
regel 540	165
regel 560	58
regel 580	122
regel 600	34
regel 620	187
regel 640	194
regel 660	240
regel 680	80
regel 700	128
regel 720	59
regel 740	149
regel 760	113
regel 780	11
regel 800	94
regel 820	50
regel 840	132
regel 860	95
regel 880	129
regel 900	58
regel 920	189
regel 940	138
regel 960	89
regel 1000	16
regel 1020	54
regel 1040	240
regel 1060	240
regel 1080	168
regel 1100	240
regel 1120	240
regel 1140	240
regel 1160	19
regel 1180	16
regel 1200	194
regel 1220	111
regel 1240	89
regel 1280	58
regel 1300	131
regel 1320	147
regel 1340	83
regel 1360	142
regel 1380	36
regel 1400	196
regel 1420	58
regel 1440	130
regel 1460	185
regel 1480	148
regel 1500	56
regel 1520	75
regel 1540	36
regel 1560	172
regel 1580	101
regel 1600	33
regel 1620	47
regel 1640	172
regel 1660	114
regel 1680	188
regel 1700	58
regel 1720	146
regel 1740	77
regel 1760	68
regel 1780	27
regel 1800	178
regel 1820	31
regel 1840	85
regel 1860	68
regel 1880	227
regel 1900	65
regel 1920	101
regel 1940	244
regel 1960	48
regel 1980	138
regel 2000	24
regel 2020	178
regel 2040	21
regel 2060	169
regel 2080	58
regel 2100	77
regel 2120	147
regel 2140	188
regel 2160	179
regel 2180	49
regel 2200	164
regel 2220	88
regel 2240	29
regel 2260	166
regel 2280	9
regel 2300	151
regel 2320	239
regel 2340	102
regel 2360	221
regel 2380	230
regel 2400	178
regel 2420	22
regel 2440	171
regel 2460	58
regel 2480	148
regel 2500	77
regel 2520	143
regel 2540	134
regel 2560	232
regel 2580	224
regel 2600	86
regel 2620	56
regel 2640	133
regel 2660	23
regel 2680	58
regel 2700	170
regel 2720	149
regel 2740	77
regel 2760	143
regel 2780	134
regel 2800	98
regel 2820	242
regel 2840	20
regel 2860	48
regel 2880	20
regel 2900	53
regel 2920	133
regel 2940	250
regel 2960	178
regel 2980	120
regel 3000	169
regel 3020	58
regel 3040	221
regel 3060	45
regel 3080	130
regel 3100	157
regel 3120	218
regel 3140	50
regel 3160	78
regel 3180	3
regel 3200	129
regel 3220	93
regel 3240	246
regel 3260	172
regel 3280	122
regel 3300	117
regel 3320	125
regel 3340	120
regel 3360	124
regel 3380	125
regel 3400	17
regel 3420	82
regel 3440	58
regel 3460	10
regel 3480	249
regel 3500	73
regel 3520	221
regel 3540	216
regel 3560	117
regel 3580	201
regel 3600	219
regel 3620	127
regel 3640	127
regel 3660	127
regel 3680	127
regel 3700	205
regel 3720	178
regel 3740	212
regel 3760	161
regel 3780	51
regel 3800	172
regel 3820	114
regel 3840	115
regel 3860	49
regel 3880	6
regel 3900	242
regel 3920	174
regel 3940	25
regel 3960	153
regel 3980	59
regel 4000	132
regel 4020	232
regel 4040	89
regel 4060	179
regel 4080	85
regel 4100	58
regel 4120	69
regel 4140	38
regel 4160	75
regel 4180	82
regel 4200	175
regel 4220	213
regel 4240	27
regel 4260	74
regel 4280	59
regel 4300	97
regel 4320	6
regel 4340	242
regel 4360	168
regel 4380	36
regel 4400	242
regel 4420	254
regel 4440	227
regel 4460	117
regel 4480	76
regel 4500	168
regel 4520	85
regel 4540	10
regel 4560	203
regel 4580	209
regel 4600	155
regel 4620	58
regel 4640	155
regel 4660	101
regel 4680	6
regel 4700	251
regel 4720	218
regel 4740	227
regel 4760	204
regel 4780	213
regel 4800	12
regel 4820	227
regel 4840	168
regel 4860	85
regel 4880	246
regel 4900	203
regel 4920	209
regel 4940	58
regel 4960	119
regel 4980	88
regel 5000	134
regel 5020	155
regel 5040	194
regel 5060	119
regel 5080	206
regel 5100	171
regel 5120	206
regel 5140	90
regel 5160	128
regel 5180	26
regel 5200	56
regel 5220	178
regel 5240	198
regel 5260	144
regel 5280	58
regel 5300	181
regel 5320	249
regel 5340	69
regel 5360	190
regel 5380	97
regel 5400	6
regel 5420	155
regel 5440	13
regel 5460	231
regel 5480	163
regel 5500	173
regel 5520	23
regel 5540	175
regel 5560	207
regel 5580	68
regel 5600	203
regel 5620	174
regel 5640	56
regel 5660	58
regel 5680	158
regel 5700	204
regel 5720	160
regel 5740	20
regel 5760	190
regel 5780	97
regel 5800	6
regel 5820	242
regel 5840	238
regel 5860	230
regel 5880	165
regel 5900	175
regel 5920	240
regel 5930	235
regel 5940	87
regel 5960	108
regel 5980	19
regel 6000	150
regel 6020	163
regel 6040	108
regel 6060	19
regel 6080	174
regel 6100	95
regel 6120	67
regel 6140	172
regel 6160	230
regel 6180	9
regel 6200	84
regel 6220	203
regel 6240	56
regel 6260	176
regel 6280	175
regel 6300	168
regel 6320	154
regel 6340	203
regel 6360	139
regel 6380	110
regel 6400	171
regel 6420	56
regel 6440	159
regel 6460	189
regel 6480	119
regel 6500	234
regel 6520	56
regel 6540	58
regel 6560	107
regel 6580	108
regel 6600	101
regel 6620	58
regel 6640	19
regel 6660	217
regel 6680	242
regel 6700	230
regel 6720	22
regel 6740	185
regel 6760	82
regel 6780	231
regel 6800	106
regel 6820	36
regel 6840	102
regel 6860	57

Vervolg Checksum Woordtraining

regel 6880	7
regel 6900	176
regel 6920	58
regel 6940	205
regel 6960	130
regel 6980	108
regel 7000	172
regel 7020	196
regel 7040	209
regel 7060	125
regel 7080	109
regel 7100	128
regel 7120	32
regel 7140	142
regel 7160	43
regel 7180	227
regel 7200	61
regel 7220	150
regel 7240	239
regel 7260	181
regel 7280	217
regel 7300	171
regel 7320	112
regel 7340	126
regel 7360	189
regel 7380	22
regel 7400	75
regel 7420	178
regel 7440	21
regel 7460	55
regel 7480	118
regel 7500	100
regel 7520	93
regel 7540	203
regel 7560	58
regel 7580	110
regel 7600	75
regel 7620	178
regel 7640	186
regel 7660	102
regel 7680	58
regel 7700	185
regel 7720	88
regel 7740	192
regel 7760	161
regel 7780	219
regel 7800	194
regel 7820	98
regel 7840	209
regel 7860	171
regel 7880	128
regel 7900	166
regel 7920	75
regel 7940	119
regel 7960	93
regel 7980	161
regel 8000	58
regel 8020	252
regel 8040	88
regel 8060	192
regel 8080	161
regel 8100	219
regel 8120	194
regel 8140	98
regel 8160	206
regel 8180	171
regel 8200	206
regel 8220	90
regel 8240	128
regel 8260	58
regel 8280	84
regel 8300	29
regel 8320	21
regel 8340	55
regel 8360	110
regel 8380	230
regel 8400	147
regel 8420	226
regel 8440	58
regel 8460	98
regel 8480	124
regel 8500	93
regel 8520	112
regel 8540	170
regel 8560	90
regel 8580	19
regel 8600	17
regel 8620	58
regel 8640	123
regel 8660	222
regel 8680	39
regel 8700	159
regel 8720	93
regel 8740	161
regel 8760	93
regel 8780	36
regel 8800	75
regel 8820	134
regel 8840	126
regel 8860	238
regel 8880	195
regel 8900	19
regel 8920	206
regel 8940	7
regel 8960	110
regel 8980	145
regel 9000	170
regel 9020	112
regel 9040	182
regel 9060	170
regel 9080	130
regel 9100	167
regel 9120	152
regel 9140	146
regel 9160	31
regel 9180	218
regel 9200	172
regel 9220	114
regel 9240	173
regel 9260	250
regel 9280	193
regel 9300	206
regel 9320	58
regel 9340	69
regel 9360	76
regel 9380	54
regel 9400	45
regel 9420	7
regel 9440	115
regel 9460	142
regel 9480	58
regel 9500	196
regel 9520	119
regel 9540	159
regel 9560	82
regel 9580	148
regel 9600	163
regel 9620	39
regel 9640	194
regel 9660	175
regel 9680	0
regel 9700	116
regel 9720	91
regel 9740	58
regel 9760	85
regel 9780	39
regel 9800	158
regel 9820	82
regel 9840	147
regel 9860	162
regel 9880	149
regel 9900	193
regel 9920	101
regel 9940	37
regel 9960	116
regel 9980	90
regel 10000	58
regel 10020	91
regel 10040	218
regel 10060	81
regel 10080	79
regel 10100	192
regel 10120	202
regel 10140	119
regel 10160	207
regel 10180	69
regel 10200	58
regel 10220	150
regel 10240	100

PRINT OUT Amiga met Rubik's Clock

Rubiks Clock

Het uurwerk dat waarschijnlijk de mensheid de laatste jaen het meeste heeft bezig gehouden in waarschijnlijk die van de Heer Rubik. Maar nu dit programma is gepubliceerd doet ook M.J. Kemps, de maker van deze computerversie van deze klok een goede gooi. Het programma is volledig muis gestuurd door middel van de pull-down menu's.

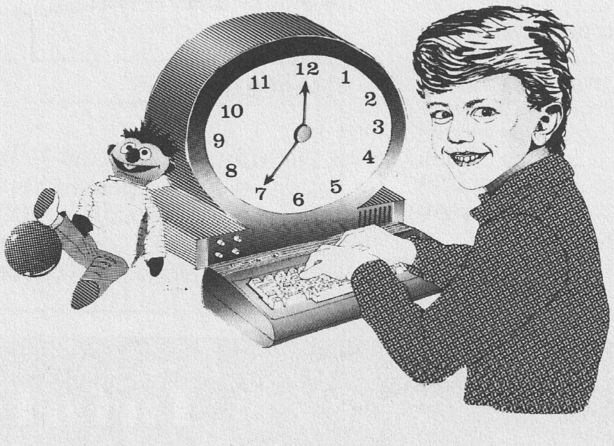
Let op de [RETURN] tussen de grote haken niet intypen maar uitvoeren. Dus gewoon op de return-toets drukken. Verder alles achter elkaar doortikken.

```
'Rubiks Clock [RETURN]
'door M.J. Kemps uit Schiedam [RETURN]
[RETURN]
CLEAR ,50000& [RETURN]
[RETURN]
Pointer: [RETURN]
FOR x=0 TO 63:READ y:POKE
20212+x,y:NEXT x [RETURN]
DATA 254,0,0,120,252,0,120
,248,248,0,113,199,240,0,97,134,224,0,65,1
28,192,6,1,128,128,7,1,192,0,0,0,248,0,0,0
,120,0,0,0,0,0,0,162,230,0,0,178,137,0,0,1
70,201,0,0,170,137,0,0,166,137,0,0,162,134
[RETURN]
[RETURN]
Startup: [RETURN]
RANDOMIZE TIMER [RETURN]
pi=3.1415926536#  [RETURN]
DIM DrawClock(6024),Hour(2,12),Button(4)
[RETURN]
clockside=1:Button(1)=1:Button(2)=1:Button
(3)=1:Button(4)=1 [RETURN]
SCREEN 2,320,216,3,1:WINDOW 2,"Rubiks
Clock",,16,2 [RETURN]
PALETTE 0,.4,.4,.4 [RETURN]
PALETTE 1,1,1,1 [RETURN]
PALETTE 2,1,0,0 [RETURN]
PALETTE 3,0,1,0 [RETURN]
PALETTE 4,1,1,0 [RETURN]
PALETTE 5,0,.5,1 [RETURN]
PALETTE 6,0,0,0 [RETURN]
PALETTE 7,0,1,1 [RETURN]
PALETTE 17,1,0,0 [RETURN]
PALETTE 18,0,0,.7 [RETURN]
PALETTE 19,1,1,1 [RETURN]
MENU OFF [RETURN]
MENU 1,0,1,"Programm":MENU 1,1,1,"Info
":MENU 1,2,1,"Quit     " [RETURN]
MENU 2,0,1,"Options":MENU
2,1,1,"Shuffle":MENU
2,2,1,"Instructions" [RETURN]
MENU 3,0,1,"":MENU 4,0,1,"":MENU
5,0,1,"" [RETURN]
ON MENU GOSUB Checkmenu: [RETURN]
[RETURN]
draw: [RETURN]
CIRCLE (206,54),33,6,,,1:PAINT
(206,54),6 [RETURN]
CIRCLE (114,54),33,6,,,1:PAINT
(114,54),6 [RETURN]
CIRCLE (114,146),33,6,,,1:PAINT
(114,146),6 [RETURN]
CIRCLE (206,146),33,6,,,1:PAINT
(206,146),6 [RETURN]
FOR a=0 TO 2*pi STEP pi/6 [RETURN]
x=160+95*COS(a) [RETURN]
y=100+95*SIN(a) [RETURN]
AREA (x,y) [RETURN]
NEXT a [RETURN]
COLOR 5,0 [RETURN]
AREAFILL [RETURN]
CIRCLE(160,100),15,1,,,1:CIRCLE(160,82),1,
4,,,1:PAINT(160,82),4 [RETURN]
CIRCLE(114,54),15,1,,,1:CIRCLE(114,36),1,4
,,,1:PAINT(114,36),4 [RETURN]
CIRCLE(160,54),15,1,,,1:CIRCLE(160,36),1,4
,,,1:PAINT(160,36),4 [RETURN]
CIRCLE(206,54),15,1,,,1:CIRCLE(206,36),1,4
,,,1:PAINT(206,36),4 [RETURN]
CIRCLE(114,100),15,1,,,1:CIRCLE(114,82),1,
4,,,1:PAINT(114,82),4 [RETURN]
CIRCLE(206,100),15,1,,,1:CIRCLE(206,82),1,
4,,,1:PAINT(206,82),4 [RETURN]
CIRCLE(114,146),15,1,,,1:CIRCLE(114,128),1
,4,,,1:PAINT(114,128),4 [RETURN]
CIRCLE(160,146),15,1,,,1:CIRCLE(160,128),1
,4,,,1:PAINT(160,128),4 [RETURN]
CIRCLE(206,146),15,1,,,1:CIRCLE(206,128),1
,4,,,1:PAINT(206,128),4 [RETURN]
CIRCLE(137,77),5,4,,,1:PAINT(137,77),4
[RETURN]
CIRCLE(183,77),5,4,,,1:PAINT(183,77),4
[RETURN]
CIRCLE(137,123),5,4,,,1:PAINT(137,123),4
[RETURN]
CIRCLE(183,123),5,4,,,1:PAINT(183,123),4
[RETURN]
FOR clock=1 TO 12 [RETURN]
CIRCLE(160,100),3,1,,,1:PAINT(160,100),1
[RETURN]
x1=INT(3*SIN((9+clock)*pi/6)+.5):y1=INT(3*
COS((9+clock)*pi/6)+.5) [RETURN]
x2=INT(160+13*COS((clock+9)*pi/6)+.5):y2=I
NT(100+13*SIN((clock+9)*pi/6)+.5)
[RETURN]
AREA(160+x1,100-y1):AREA(x2,y2):AREA(160-x
1,100+y1):COLOR 1,5:AREAFILL [RETURN]
GET(145,85)-(175,115),DrawClock((clock-1)*
502) [RETURN]
PAINT(160,100),5 [RETURN]
NEXT clock [RETURN]
COLOR 4,0:LOCATE 23,1:PRINT " Side 1
":LOCATE 23,32:PRINT " Side 2 " [RETURN]
LINE(4,173)-(60,186),4,b:LINE(252,173)-(30
8,186),4,b [RETURN]
GOSUB Shuffle [RETURN]
MENU ON [RETURN]
[RETURN]
Puzzle: [RETURN]
IF MOUSE(0)=0 THEN Puzzle [RETURN]
x=MOUSE(1):y=MOUSE(2) [RETURN]
IF x4 AND x< AND y173 AND y‡ THEN
clockside=1 [RETURN]
IF x252 AND x4 AND y173 AND y‡ THEN
clockside=2 [RETURN]
Change=0 [RETURN]
IF x132 AND xÄ AND y72 AND yR THEN
Change=1 [RETURN]
```

```
IF x178 AND x¶ AND y72 AND yR THEN
Change=2 [RETURN]
IF x132 AND xÄ AND y118 AND yÇ THEN
Change=3 [RETURN]
IF x178 AND x¶ AND y118 AND yÇ THEN
Change=4 [RETURN]
IF Change=1 AND clockside=1 OR Change=2
AND clockside=2 THEN
Button(1)=-Button(1)+3 [RETURN]
IF Change=2 AND clockside=1 OR Change=1
AND clockside=2 THEN
Button(2)=-Button(2)+3 [RETURN]
IF Change=3 AND clockside=1 OR Change=4
AND clockside=2 THEN
Button(3)=-Button(3)+3 [RETURN]
IF Change=4 AND clockside=1 OR Change=3
AND clockside=2 THEN
Button(4)=-Button(4)+3 [RETURN]
turn=0:direction=0 [RETURN]
IF x81 AND y6 AND xy+60 THEN
turn=1:direction=-1 [RETURN]
IF xr AND y21 AND xy+60 THEN
turn=1:direction=1 [RETURN]
IF x206 AND y21 AND xy+260 THEN
turn=2:direction=-1 [RETURN]
IF x AND y6 AND x-y+260 THEN
turn=2:direction=1 [RETURN]
IF xr AND yø AND x-y+260 THEN
turn=3:direction=-1 [RETURN]
IF x81 AND y146 AND xy+260 THEN
turn=3:direction=1 [RETURN]
IF x AND y146 AND xy+60 THEN
turn=4:direction=-1 [RETURN]
IF x206 AND yø AND xy+60 THEN
turn=4:direction=1 [RETURN]
IF clockside=2 THEN
direction=-direction:turn=3-turn [RETURN]
IF turn THEN turn=turn+4 [RETURN]
IF Button(turn)=Button(1)THEN
Hour(1,1)=Hour(1,1)+direction [RETURN]
IF Button(turn)=Button(2)THEN
Hour(1,3)=Hour(1,3)+direction [RETURN]
IF Button(turn)=Button(3)THEN
Hour(1,7)=Hour(1,7)+direction [RETURN]
IF Button(turn)=Button(4)THEN
Hour(1,9)=Hour(1,9)+direction [RETURN]
IF Button(turn)=1 THEN [RETURN]
Hour(1,5)=Hour(1,5)+direction [RETURN]
IF Button(1)=1 OR Button(2)=1 THEN
Hour(1,2)=Hour(1,2)+direction [RETURN]
IF Button(1)=1 OR Button(3)=1 THEN
Hour(1,4)=Hour(1,4)+direction [RETURN]
IF Button(2)=1 OR Button(4)=1 THEN
Hour(1,6)=Hour(1,6)+direction [RETURN]
IF Button(3)=1 OR Button(4)=1 THEN
Hour(1,8)=Hour(1,8)+direction [RETURN]
ELSE [RETURN]
Hour(2,5)=Hour(2,5)-direction [RETURN]
IF Button(1)=2 OR Button(2)=2 THEN
Hour(2,2)=Hour(2,2)-direction [RETURN]
IF Button(1)=2 OR Button(3)=2 THEN
Hour(2,6)=Hour(2,6)-direction [RETURN]
IF Button(2)=2 OR Button(4)=2 THEN
Hour(2,4)=Hour(2,4)-direction [RETURN]
IF Button(3)=2 OR Button(4)=2 THEN
Hour(2,8)=Hour(2,8)-direction [RETURN]
END IF [RETURN]
GOSUB SetClocks [RETURN]
release: [RETURN]
IF MOUSE (0)=-1 THEN release [RETURN]
GOTO Puzzle [RETURN]
[RETURN]
SetClocks: [RETURN]
IF clockside=1 THEN PALETTE
5,0,.5,1:ELSE PALETTE 5,0,0,1 [RETURN]
Hour(2,1)=12-Hour(1,3):Hour(2,3)=12-Hour(1
,1):Hour(2,7)=12-Hour(1,9):Hour(2,9)=12-Ho
ur(1,7) [RETURN]
FOR side=1 TO 2:FOR clock=1 TO 12
[RETURN]
IF Hour(side,clock) THEN
Hour(side,clock)=Hour(side,clock)+12
[RETURN]
IF Hour(side,clock)12 THEN
Hour(side,clock)=Hour(side,clock)-12
[RETURN]
NEXT clock,side [RETURN]
PUT(99,39),DrawClock((Hour(clockside,1)-1)
*502),PSET [RETURN]
PUT(145,39),DrawClock((Hour(clockside,2)-1
)*502),PSET [RETURN]
PUT(191,39),DrawClock((Hour(clockside,3)-1
)*502),PSET [RETURN]
PUT(99,85),DrawClock((Hour(clockside,4)-1)
*502),PSET [RETURN]
PUT(145,85),DrawClock((Hour(clockside,5)-1
)*502),PSET [RETURN]
PUT(191,85),DrawClock((Hour(clockside,6)-1
)*502),PSET [RETURN]
PUT(99,131),DrawClock((Hour(clockside,7)-1
)*502),PSET [RETURN]
PUT(145,131),DrawClock((Hour(clockside,8)-
1)*502),PSET [RETURN]
PUT(191,131),DrawClock((Hour(clockside,9)-
1)*502),PSET [RETURN]
IF Button(1)=1 AND clockside=1 OR
Button(2)=2 AND clockside=2 THEN
bcolor=4:ELSE bcolor=5 [RETURN]
CIRCLE(137,77),5,bcolor,,,1 [RETURN]
IF Button(2)=1 AND clockside=1 OR
Button(1)=2 AND clockside=2 THEN
bcolor=4:ELSE bcolor=5 [RETURN]
CIRCLE(183,77),5,bcolor,,,1 [RETURN]
IF Button(3)=1 AND clockside=1 OR
Button(4)=2 AND clockside=2 THEN
bcolor=4:ELSE bcolor=5 [RETURN]
CIRCLE(137,123),5,bcolor,,,1 [RETURN]
IF Button(4)=1 AND clockside=1 OR
Button(3)=2 AND clockside=2 THEN
bcolor=4:ELSE bcolor=5 [RETURN]
CIRCLE(183,123),5,bcolor,,,1 [RETURN]
RETURN [RETURN]
[RETURN]
Checkmenu: [RETURN]
menuid=MENU(0):menuitem=MENU(1) [RETURN]
ON menuid GOSUB Programm,Options
[RETURN]
RETURN [RETURN]
[RETURN]
Programm: [RETURN]
ON menuitem GOSUB Info,Quit [RETURN]
RETURN [RETURN]
[RETURN]
Info: [RETURN]
WINDOW 3,"Info",(74,50)-(248,138),0,2
[RETURN]
```

```
MENU OFF:COLOR 5,1:CLS [RETURN]
PRINT:PRINT "  Rubiks Clock  v1.0"
[RETURN]
COLOR 6,1:PRINT:PRINT "  Created in
1989 by" [RETURN]
COLOR 2,1:PRINT "        M.J. Kemps"
[RETURN]
COLOR 6,1:PRINT:PRINT "     Designed
for" [RETURN]
COLOR 5,1:PRINT "    Commodore INFO"
[RETURN]
COLOR 6,4:LOCATE 10,11:PRINT "OK"
[RETURN]
LINE (77,69)-(98,82),2,b [RETURN]
Checkmouse: [RETURN]
IF MOUSE(0)=0 THEN Checkmouse [RETURN]
IF MOUSE(3)P OR MOUSE(3)95 THEN
Checkmouse [RETURN]
IF MOUSE(4)H OR MOUSE(4)79 THEN
Checkmouse [RETURN]
COLOR 1,6:LOCATE 10,11:PRINT "OK"
[RETURN]
toetslos: [RETURN]
IF MOUSE(0)0 THEN toetslos [RETURN]
WINDOW CLOSE 3:MENU ON [RETURN]
RETURN [RETURN]
[RETURN]
Quit: [RETURN]
WINDOW 3,"Quit",(80,50)-(240,138),0,2
[RETURN]
MENU OFF:COLOR 5,1:CLS [RETURN]
LOCATE 3,4:PRINT "Are you sure ?"
[RETURN]
COLOR 6,4:LOCATE 9,3:PRINT " Quit
":LOCATE 9,13:PRINT "Cancel" [RETURN]
LINE (13,61)-(66,74),2,b:LINE
(93,61)-(146,74),2,b [RETURN]
Checkmouse2: [RETURN]
IF MOUSE(0)=0 THEN Checkmouse2 [RETURN]
IF MOUSE(3)@ AND MOUSE(3)15 AND
MOUSE(4)63 AND MOUSE(4)H THEN Quit1
[RETURN]
IF MOUSE(3)É AND MOUSE(3)95 AND
MOUSE(4)63 AND MOUSE(4)H THEN Cancel
[RETURN]
GOTO Checkmouse2 [RETURN]
Quit1: [RETURN]
COLOR 1,6:LOCATE 9,3:PRINT " Quit "
[RETURN]
wacht1: [RETURN]
IF MOUSE(0)0 THEN wacht1 [RETURN]
WINDOW CLOSE 3:WINDOW CLOSE 2:SCREEN
CLOSE 2 [RETURN]
MENU RESET [RETURN]
END [RETURN]
Cancel: [RETURN]
COLOR 1,6:LOCATE 9,13:PRINT "Cancel"
[RETURN]
wacht2: [RETURN]
IF MOUSE(0)0 THEN wacht2 [RETURN]
WINDOW CLOSE 3:MENU ON [RETURN]
RETURN [RETURN]
[RETURN]
Options: [RETURN]
ON menuitem GOSUB Shuffle,Instructions
[RETURN]
RETURN [RETURN]
[RETURN]
Shuffle: [RETURN]
FOR side=1 TO 2:FOR clock=1 TO 12
[RETURN]
Hour(side,clock)=INT(12*RND(1)+1)
[RETURN]
NEXT clock,side [RETURN]
GOSUB SetClocks [RETURN]
RETURN [RETURN]
[RETURN]
Instructions: [RETURN]
WINDOW 4,"instructions",,0,2 [RETURN]
MENU OFF:COLOR 7,6:CLS [RETURN]
PALETTE 2,.8,0,.8:a=MOUSE(0) [RETURN]
PRINT :PRINT "All you have to do is to
get all clocks" [RETURN]
PRINT "at 12 o'clock by turning the
wheels and" [RETURN]
PRINT "changing the buttons." [RETURN]
PRINT :PRINT "Don't forget the other
side!" [RETURN]
PRINT:COLOR 2,6 [RETURN]
PRINT "Control:" [RETURN]
PRINT :PRINT "Changing Buttons:
";:COLOR 3,6:PRINT "Simply klick on
them" [RETURN]
COLOR 2,6:PRINT :PRINT "Turning wheels:
";:COLOR 3,6:PRINT "Klick on the side
of" [RETURN]
PRINT "                          the direction
you" [RETURN]
PRINT "                          want to turn"
[RETURN]
COLOR 2,6:PRINT :PRINT "Change side:
";:COLOR 3,6:PRINT "Klick on Side X"
[RETURN]
COLOR 4,6:LOCATE 20,10:PRINT "Press
left mousebutton" [RETURN]
klick1: [RETURN]
IF MOUSE(0)=0 THEN klick1 [RETURN]
WINDOW CLOSE 4:PALETTE 2,1,0,0:MENU ON
[RETURN]
RETURN [RETURN]
```

Einde listing Rubik's Clock



Niet zo lange tijd geleden, in een achter een groot water gelegen land, werd een machine uitgevonden. Het karakter van deze machine had een bijna menselijk,jawel bijna vrouwelijk, voorkomen. De maker van de machine zag dit en gaf het in een poëtische bui een naam. Ze was zijn steun en toeverlaat in moeilijke tijden, ze was er wanneer hij haar nodig had, kortweg, ze was zijn vriendin, zij was zijn 'Amiga'.

Binnenin AmigaDOS (7)

Een zeer poëtisch begin. U zult zich afvragen, wat was nou de bedoeling van het bovenstaande. Het antwoord hierop luidt: GEEN!

Nu verder met onze cursus. De vorige keer waren we gebleven bij de behandeling van een aantal handlers en devices. Deze keer gaan we verder met de commando's die al bestaan. Ze zijn echter enigszins aangepast, dus het is weer de moeite waard de commando's opnieuw te bekijken. We hanteren de zelfde beschouwingsmethode als in de vorige afleveringen. De commando's zullen alfabetisch ten tonele worden geleid.

Vernieuwde commando's

Addbuffers

Op het eerste gezicht lijkt hier niet zoveel in veranderd. Schijn bedriegt. Onder de oppervlakte van onze commando-oceaan is weer heel wat aangepast, immers **Addbuffers** moest samen kunnen werken met het FastFileSystem. Goed, de gemiddelde gebruiker zal hiervan niets merken. Wat merkt de gebruiker wel? Het is nu mogelijk geworden te bepalen in welk geheugengebied de buffers worden geplaatst. Door middel van het 'BUFMEMTYPE' keyword wordt bepaald in welk geheugengebied de buffer wordt geplaatst. (Zie tabel 1)

Code	Geheugen
0,1	Al naar gelang welk geheugen vrij is
2,3	CHIP-memory
4,5	FAST-memory

Tabel 1. Gebruikte geheugen voor buffer

Heeft u bijvoorbeeld een Amiga 500 zonder geheugenuitbreiding, dan zullen de mogelijkheden 4 en 5 wegvallen, immers u beschikt niet over FAST memory.



Assign

De Workbench v1.3 **Assign** vermeldt nu alle vermeldingen met meer dan drie letters zonder fouten. Scheelt weer heel wat zoekwerk naar de plaats waar de de fout eigenlijk ingeslopen was, nietwaar!
Assign ondersteunt ook weer een extra optie. Als u intypt **Assign EXISTS [naam]**, dan zal het commando **Assign** de vermeldingen-lijst doorzoeken naar de achter EXISTS vermelde naam. Als de naam voorkomt, zal **Assign** 0 teruggeven. Komt de opgegeven naam niet voor, dan zal 5 (=WARN) teruggegeven worden. Hiermee kan dan met behulp van een **IF WARN** constructie op een bepaalde vermelding getest worden. Zou de uitkomst gelijk aan 5 (=WARN) zijn, dan zou alsnog deze vermelding 'geassigned' kunnen worden. Een voorbeeld hiervan,

```
Assign EXISTS FONTS:
If WARN
  Assign FONTS: df0:fonts
Endif
```

Copy

Het commando **Copy** is in WB1.3 uitgebreider geworden. Er zijn meerdere opties bijgekomen. Het merendeel van deze opties dankt zijn bestaansrecht aan de nieuwe statusbits. De volgende opties zijn er bijgekomen,
BUF : met deze optie is het mogelijk het aantal, 512 bytes grote, buffers te regelen welke het Copy commando tijdens het kopieer-proces gebruikt.
ALL : hiermee is het mogelijk om meerdere files of een gehele directory te kopiëren. Het is soortgelijk aan het 'pattern matching', u weet wel, het gebruik van de '#?' tekens.
DATE : Copy zal nu de datum waarop de file gecreerd is ook kopiëren.
NOPRO: door gebruik van deze optie worden de statusbits niet mee-gekopieerd naar de andere file.
COM : het is mogelijk in DOS een commentaar aan een file te hechten. Dit wordt normaal gezien niet mee gekopieerd. Door gebruik van de optie 'COM' zal het commentaar ook meegenomen worden naar de nieuwe file.
CLONE : als laatste optie tenslotte, de optie CLONE. Door gebruik van deze optie worden alle eigenschap-

pen, dus datum, statusbits, etcetera, gekopieerd naar de nieuwe file.

De syntax van **Copy**:

```
COPY [FROM] source [TO] destination [ALL][QUIET][BUF=][CLONE][DATE][NOPRO][COM]
```

Een voorbeeld:

```
COPY :devs/printers TO df1:devs/printers ALL
```

Als er een directory wordt vermeld in de filenaam en deze direcory bestaat niet, dan zal **Copy** deze eerst creëren. Dit voorkomt weer extra werk, nietwaar!
Tenslotte is er dan nog de mogelijkheid om de tekens "" (2 keer aanhalingsteken) te gebruiken ter aanduiding van de actuele directory. Dus wilt u in de zelfde directory kopiëren, dan is het niet meer nodig deze directory naam twee keer te vermelden. Eén keer en het gebruik van de twee aanhalingstekens is voldoende geworden.

Dir

Ook in dit commando zijn verschillende zaken veranderd. Voor alle afkortingen zijn opties gecreëerd. U kent de letters ongetwijfeld nog wel, A, I en D. Door de optie OPT te gebruiken kon u een van deze letters toevoegen waardoor **Dir** een bepaalde optie uitvoerde. Het gebruik van OPT is niet langer nodig. Simpelweg een van onderstaande opties intypen en voila, het gewenste staat op het scherm.
De nieuwe opties luiden als volgt;
ALL :het synoniem voor A
INTER:het synoniem voor I
DIRS :het synoniem voor D
COM :de eerste echt nieuwe voor **Dir**. Het is hiermee mogelijk geworden een commando in te voeren. Het keyword COM of C moet dan wel in de inter-actieve mode (INTER) gebruikt worden. Typ achter het vraagteken een C en vervolgens daarachter het commando dat u uit wilt voeren. (Zie voorbeeld).
De syntax van **Dir**:
DIR device [OPT A|I|D] [ALL][DIRS] [INTER[COM]]
Een voorbeeld:
DIR df0:s INTER
Op het scherm kan bijvoorbeeld komen te staan:

```
s(dir)
Startup-Sequence StartupII
? C"type s/startup-sequence"
```

Hieruit blijkt wel weer dat **Dir** weer wat krachtiger en gebruikersvriendelijker geworden is.

DiskDoctor

Diskdoctor's interne routines zijn enigszins aangepast. Het controleert eerst of er genoeg geheugen is voordat er begonnen wordt met de kopieeroperatie. Natuurlijk werkt het ook samen met het nieuwe filesysteem, FastFileSystem. Als u van deze mogelijkheid gebruik wilt maken dan moet u niet vergeten in de MountList, achter het keyword DOSTYPE het juiste 'dostype' op te geven. De volgende vermelding in de MountList zorgt ervoor dat het FastFileSystem gebruikt wordt:

```
DosType = 0x444f5301
```



Typ dit regeltje over het oude dostype heen, of als het er in zijn geheel nog niet staat, typ het dan in binnen de vermelding voor het specifieke device.

Echo

Echo kent drie extra opties. Alle drie hebben ze invloed op de tekst die achter **Echo** ingetypt wordt. De opties zijn:
NOLINE :zoals de optie al zegt; er wordt geen RETURN gegeven achter de tekenreeks die **Echo** uitvoert.
FIRST nn : de eerste nn tekens worden niet uitgevoerd. Dus voor nn moet het aantal tekens ingevuld worden welke u niet afgeprint wilt zien.
LEN nn : nu gaat het om het aantal tekens dat u afgebeeld wilt zien, gerekend vanaf het einde.
De syntax van **Echo**:

```
ECHO string [NOLINE][FIRST
     nn][LEN nn]
```

Een voorbeeld:

```
ECHO "Dit is een test"LEN 4
```

Op het beeldscherm zal dan afgebeeld worden:

```
test
```

Execute

Ook bij **Execute** hebben zich weer de nodige veranderingen voorgedaan. Als **Execute** als resident-commando in de SHELL omgeving werkt zal men zich in de zevende hemel wanen. De commando's worden dan direct uitgevoerd. Ook de batch-bestanden waarvan het zogenaamde 'script' bit gezet is, u weet wel één van de statusbits, laden eerst het **Execute** commando. **Execute** maakt gebruik van het virtuele device T: als deze 'geassigned' is.

Zo niet, dan maakt het gebruik van de T-directory. Goed, we geven toe, geen schokkende veranderingen, wat de gebruiker aangaat. Eén ding is wel van het belang: Het gebruik van **Execute** in de SHELL omgeving werkt een stuk prettiger dan voorheen. Even een voorbeeld. Stel u maakt een batchbestand dat de naam 'compact' heeft. Geef nu op:
PROTECT compact +s
Het bestand 'compact' zal nu een gezet scriptbit hebben. Typ nu in:
compact
Heee, hoe wonderlijk. Geen foutmeldingen die schijnbaar doelloos over het scherm dwalen, nee het batchbestand wordt gewoon uitgevoerd alsof het hier om een com-mando gaat. Dit geeft nu de kracht van **Execute** in de SHELL omgeving aan.

Format

Eigenlijk is het wel logisch dat dit commando enige veranderingen is ondergaan, immers het moet samen kunnen werken met het FastFileSystem (FFS).
Welke opties zijn nu nieuw?
QUICK :hiermee wordt alleen het 'Root Block', het 'Boot Block' en de 'BitMap Blocks' opnieuw geformatteerd. Zeer handig voor de harddisk bezitters.
FFS : deze optie geeft aan dat het device onder FFS geformatteerd moet worden.
NOFFS : tja, het spreekt eigenlijk wel voor zich. Dit geeft aan dat het oude FileSystem gebruikt moet worden.

De opties FFS en NOFFS kunnen worden gebruikt om het in de MountList vastgestelde FileSystem, dit is datgene wat u, of de programmeur, opgegeven had achter het keyword DosType, te 'overriden' zoals de Engelsen het zo mooi kunnen zeggen. U geeft dus in de MountList een FileSystem type op, bijvoorbeeld u gaf op dat het gewone FileSystem gebruikt moest worden. Echter, u wilt het device nu onder het FastFileSystem formatteren. Geef dan de optie FFS mee en het device zal onder het FastFileSystem geformatteerd worden.
Mischien nog even ten overvloede: als u het device onder het oude filesystem wilt gebruiken dan is niet nodig het DosType keyword in de MountList te gebruiken. Wilt u het FastFileSystem wel gebruiken, dan zult u achter DosType, 0x444f5301 moeten zetten.
De syntax van **Format**:

```
FORMAT DRIVE device NAME naam [NOICONS]
[QUICK][FFS][NOFFS]
```

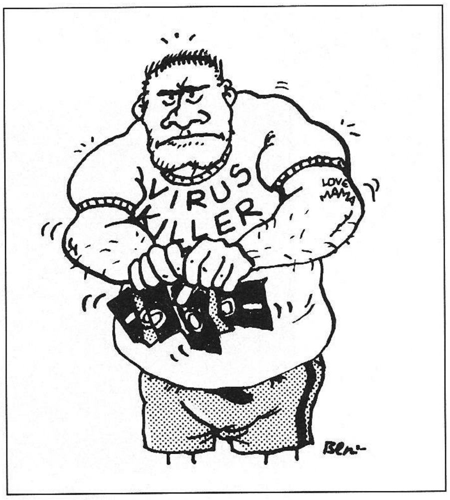
Een voorbeeld:

```
FORMAT DRIVE RAD: NAME RAMBO FFS
```

Dit voorbeeld zorgt ervoor dat de reset-vaste RAM disk geformatteerd wordt onder FFS. Het is echter wel nodig een paar zaken hiervoor in de MountList te veranderen. Zie hiervoor Commodore INFO nr. 2, bladzijde 65, Tips en Trucks.

Info

Hier hebben zich niet zoveel veranderingen voorgedaan. Een kleine verandering, de afgeprinte tekst van **Info** past zich automatisch aan, aan de langste schijf (volume) naam.



Er is wel een nieuwe optie bijgekomen. Door toevoeging van een device vermelding is het mogelijk een overzicht te krijgen van één device.
De syntax van **Info**:

```
INFO [device]
```

Install

Ook Commodore is in de aanval tegen virussen. Al geruime tijd kan men 'viruswatchers', '-protectors', etcetera in de handel verkrijgen. Met behulp van het **Install** commando kan men op gezette tijden de schijven controleren of er zich virussen op genesteld hebben. De optie CHECK zorgt ervoor dat **Install** de bootsector van de te controleren schijf vergelijkt met een standaard patroon. Komt dit niet met elkaar overeen, dan zal **Install** een waarschuwing geven en de WARN-flag zal gezet worden. Raak niet meteen in paniek bij een eventuele melding, veel commerciële programmaturen hebben ook eigen, dus een afwijkende, bootsector.
Een tweede nieuwe optie is NOBOOT. Hiermee kunt u een bootsector wel opnieuw installeren; echter, de schijf wordt dan als niet 'bootbaar' gemerkt. U kunt dus met een op deze wijze geïnstalleerde schijf niet opstarten.
De syntax van **Install**:

```
INSTALL [DRIVE]
[df0|df1|df2| df3]: [NO-
BOOT][CHECK]
```

Een voorbeeld:

```
INSTALL DRIVE df0: CHECK
IF WARN
   ECHO "Op deze schijf zou
wel een virus genesteld kun-
nen zijn."
ENDIF
```

Join

Heel kort. **Join** kende de optie TO al. Hiervoor is er een synoniem bij gekomen. In plaats van TO mag u ook AS gebruiken.

List

Deze zal niet zo kort zijn als het voorgaande commando. In **List** is een aanzienlijk aantal zaken veranderd. Ten eerste laat **List** de nieuwe statusbits H (Hidden), S (Script), P (Pure) en A (Archive) zien.
Het is nu ook mogelijk geworden om het zogenaamde 'pattern matching', u weet wel, het gebruik van de '#?' tekens, toe te passen bij **List**. U geeft de eerste letters van een filenaam op en vervolgens de tekens '#?'. **List** zal nu alle files afbeelden die met deze letters beginnen.
Ook bij **List** is weer een aanzienlijk aantal nieuwe opties bijgekomen.
BLOCK : de filegrootte wordt afgebeeld in het aantal blokken in plaats van het aantal bytes.
NOHEAD : de meldingen die tevoorschijn komen aan het begin en het eind van de **List**-operatie, dus de directory plus datum-vermelding aan het begin en het aantal files plus directories aan het einde, worden onderdrukt.
FILES : alleen de files en hun gegevens worden afgebeeld.
DIRS : alleen de directories en hun gegevens worden afgebeeld.

Eén keyword hebben we nog niet besproken. Dit is het keyword **LFORMAT**. Het is hiermee mogelijk gewor-

den de uitvoer van **List** naar eigen wens te regelen.
Door de nu ontstane uitvoer om te leiden naar een apart bestand kunt u op eenvoudige wijze een batchbestand creëren. Even een voorbeeld:

```
LIST ram:bestand LFORMAT
"protect %S -d"
```

Dit voorbeeld zorgt ervoor dat een batchfile wordt gemaakt met de naam 'bestand'. Ze is terug te vinden op de ram-disk. Dit batchbestand zorgt ervoor van alle files in de directory waarop dit bestand wordt losgelaten, de delete-bit wordt gewist. Nu nog even een opmerking omtrent het tussenvoegen van de filenamen. Dit doet u met behulp van '%S'.
'%S' wordt in het batchbestand vervangen door de filenaam. Vermeldt u 2 keer %S, dan zal de **List**-operatie twee keer op de file uitgevoerd worden. Vermeldt u 3 keer %S dan zal **List** dit vervangen voor de directoryvermelding(en) (path), en twee keer de filenaam. Tenslotte is er nog de mogelijkheid van 4 keer %S. Dit wordt vervangen door de path, de filenaam, de path en wederom de filenaam.
De syntax van **List**:
LIST device|directory|pattern [PAT] pattern [KEYS][NODATES][TO filenaam][S substring][SINCE datum][UPTO datum] [QUICK][BLOCK][NOHEAD][FILES] [DIRS] [LFORMAT="string"]

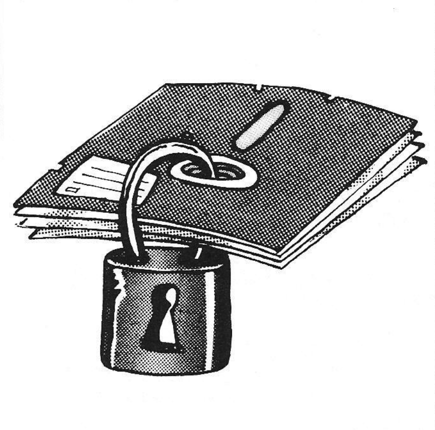
Mount

Niet verwacht, dat hieraan nog iets viel te verbeteren. Niets is minder waar! Was het voorheen mogelijk alleen devices vanuit te MountList te 'mounten', tegenwoordig kunt u elke ASCII file hiervoor gebruiken. Deel aan **Mount** mee uit welk bestand hij de gegevens moet halen en klaar is Kees...of Jan...of Piet......ho.STOP. En klaar bent u! Gebruik hier dan wel de optie FROM voor.
Een voorbeeld:
MOUNT rad: FROM df0:AltMntList

Dit voorbeeld zorgt ervoor dat de gegevens van het device 'RAD:' worden gezocht in het bestand 'AltMntList' welke op drive DF0: moet staan.

NewCli

Door middel van de optie FROM was het mogelijk automatisch bij het opstarten van een nieuwe CLI een batchbestand uit te laten voeren. In WB1.3 voert **NewCli**, iedere keer als het commando uitgevoerd wordt, een batchbestand uit. Dit wordt genoemd, het 'default startup-file'. Deze default startup-file heet 'CLI-startup' en is terug te vinden in de S-directory. Wilt u desondanks toch uw eigen batchbestand uit laten voeren, maak dan gebruik van de optie FROM en vermeld hierachter de naam van het bestand.



Path

Ook bij **Path** is er een nieuwe optie bijgekomen. De optie, genaamd QUIET, zorgt ervoor dat **Path** niet meer met de hinderlijke requesters tevoorschijn komt als **Path** haar lijst aan het doorzoeken is. Even een voorbeeld ter verduidelijking.
Stel, uw 'path' (wie weet hier een fatsoenlijk Nederlands woord voor!!!!) luidt als volgt.
SYS1:c
RAD:
RAM:
SYS2:c
SYS2:bin
Als nu SYS1 niet in een van uw drives zit, dan zal **Path** alleen de volumenaam weergeven. Ze zal de directorynaam weglaten. Het voordeel van de optie QUIET is dat er nu geen requesters meer tevoorschijn zullen komen.

De syntax van **Path**:
PATH directory-naam [SHOW][ADD][RESET] [QUIET]
Een voorbeeld:
PATH df1:bin ADD

Prompt

Ook dit verhaaltje zal niet te lang duren, er is vrij weinig veranderd. **Prompt** is in staat, tezamen met de SHELL-omgeving, het actuele device en -directories, te vermelden. Ook kunt u met **Prompt** het CLI-nummer af laten beelden. Dit wist u, als trouwe volger van deze cursus echter al. Deze twee eigenschappen kunt u door **Prompt** uit laten voeren door in de promptstring zekere codes te gebruiken. Welke zijn dit?
%N : dit wordt door **Prompt** vervangen door het actuele CLI-nummer. Deze mogelijkheid bestond al in WB1.2
%S : dit wordt door **Prompt** vervangen door de actuele directory en het actuele device.
De syntax van **Prompt** luidt als volgt:
PROMPT promptstring

Protect

Ook hierbij waren de nieuwe statusbits een grote aanleiding tot verandering. Immers, de statusbits moeten op enigerlei wijze beïnvloed kunnen worden. Deze functie wordt vervult door **Protect**. Ook de nieuwe statusbits moeten weer door **Protect** kunnen worden beïnvloed, dus verandering was noodzakelijk. Wat is er nu anders geworden? Het 'setten' of 'resetten' van de statusbits gebeurt door middel van twee opties, namelijk ADD en SUB. Voor beide opties zijn ook afkortingen beschikbaar. Voor ADD is er '+' en voor SUB is er '-'.
De syntax van **Protect**:
PROTECT [FILE] filenaam [FLAGS] [+][-] status bits [ADD][SUB]
Een voorbeeld:

PROTECT rad:daphne.c -wd

De file kan door deze actie niet meer beschreven en gewist worden. Het is de bedoeling dus om de gewenste actie, ADD(+), SUB(-), te noteren en vervolgens moet aangegeven welke statusbits beïnvloed moeten worden.

Tot Slot

In deze aflevering hebben we ons grotendeels weer gehouden aan een kale opsomming van de commando's en hun grotendeels nieuwe eigenschappen. We hebben de opzet een klein beetje gewijzigd, dat wil zeggen, in eerste instantie van elk commando een beschrijving te geven, vervolgens waar nodig een syntax beschrijving en een voorbeeld te geven.
In de volgende aflevering gaan we de AmigaDOS cursus afsluiten. We geven de laatste commando's en we gaan nog even wat dieper in op de nieuwe Preferences. Dan rest ons niets meer dan u nog hier vandaan prettige experimenteer uren met AmigaDOS toe te wensen en denk erom, een NEC MultiSync II is te duur en zet uw koffiekopje in vredesnaam niet naast het toetsenbord!!!

Tips & Trucs

Timer

AmigaBasic bevat o.a. de functie TIMER. Volgens het manual telt deze functie het aantal seconden, dat verlopen is sinds middernacht (00:00:00 uur). Dit gebeurt echter niet goed. De TIMER zou moeten tellen van 0 (= 00:00:00) t/m 86.399 (23:59:59). Ik geef gemakshalve alleen integer waarden. Er gebeurt echter iets geheel anders, zie onderstaand overzicht.
Dit komt waarschijnlijk, omdat er ergens een teller te klein bemeten is. Het is vooral lastig als je de benodigde tijd wilt meten van een bepaald programma-onderdeel m.b.v. de TIMER-functie. Je kunt zeer verrassende uitslagen krijgen. Bedenk, dat niet iedereen een real-time clock heeft. Bij diegenen, die wel een real-time clock hebben is deze nog lang niet altijd geactiveerd. M.a.w. veelal is bij spelletjes e.d. de systeem-tijd voor iedereen een volkomen geheim. Het kan net zo goed 00:01:15 zijn, als 18:12:45, als 23:59:34.

Hierachter schets ik een aantal situaties, steeds met een van de mogelijke oplossingen (er zijn vaak wel andere oplossingen te bedenken, die alle even goed, mogelijk zelfs efficiënter zijn).

° 1. U wilt gewoon precies het aantal seconden na 00:00:00 weten.

```
GOSUB seconden : PRINT  USING
"#####.####"; sec#
seconden:
sec#=TIMER
IF VAL(LEFT$(TIME$,2))17 AND
sec#ü# THEN sec#=sec#+65536#
RETURN
```

Noot: Als u minder nauwkeurig maar sneller wilt zijn, dan moet u alle double-precisions wijzigen in single-precisions of in long-integers. Short-integers kan niet, want deze kunnen niet de max. waarde van 86400 bevatten. Ook dient u het PRINT-statement overeenkomstig aan te passen (minder decimalen in het USING-gedeelte).

° 2. U wilt alleen het verschil weten tussen 2 tijdstippen. U weet zeker, dat het verschil kleiner is dan 5 uur 47 min 44 sec (20.864 sec).

```
s1#=TIMER
-----
-----
-----
s2#=TIMER : GOSUB verschil :
PRINT USING "#####.####";
s3#
```

verschil:

```
s3#=s2#-s1#
IF s3# THEN s3#=s3#+20864#
   ELSE RETURN
IF s3# THEN s3#=s3#+44672#
RETURN
```

° 3. U wilt alleen het verschil weten tussen 2 tijdstippen. U weet NIET zeker, of het verschil kleiner is dan 5 uur 47 min 44 sec (20.864 sec).

```
CALL seconden(s1#)
-----
-----
-----

CALL seconden(s2#)
s3#=s2#-s1# : IF s3# THEN
s3#=s3#+86400#
PRINT USING "#####.####";
s3#
-----
-----
END

SUB seconden(s#) STATIC
s#=TIMER
IF VAL(LEFT$(TIME$,2))17
0AND s#ü# THEN
s#=s#+65536#
END SUB
```

Systeem-Tijd	Aantal sec na 00:00:00	Waarde die de TIMER geeft
00:00:00	0	0
00:00:01	1	1
00:00:02	2	2
: :	--.--	--.--
18:12:58	65.578	65.578
18:12:59	65.579	65.579
18:13:00	65.580	44
18:13:01	65.581	45
18:13:02	65.582	46
: :	--.--	--.--
23:59:58	86.398	20.862
23:59:59	86.399	20.863
00:00:00	0	0
00:00:01	1	1



Amiga

Interpreter



Hieronder komt een aantal TIPS, die van belang kunnen zijn voor de SNELHEID van uw BASIC-programma. Er moet hierbij onderscheid gemaakt worden tussen twee omgevingen: de basic-INTERPRETER en de basic-COMPILER. Ik denk dat voor een aantal onder u niet alles even nieuw of opzienbarend is, maar voor velen onder u zal er toch wat te leren vallen.

Tip-1 (Interpreter).

Zet zo weinig mogelijk commentaar in uw programma, hoe minder commentaar en/of commentaarregels, hoe sneller uw programma. En als u dan TOCH commentaar wilt vermelden, denk dan aan het volgende:

° Houd het commentaar zo kort mogelijk.
° Plaats het commentaar zoveel mogelijk naar links
° Plaats het commentaar niet in een vaak te doorlopen loop,
(maar voor de loop of na de loop.)

Tip-2 (Interpreter)

Zet zoveel mogelijk BASIC-statements op 1 regel, uiteraard gescheiden door een ":". Een regel kan max 255 characters bevatten (bij de 256-ste komt de melding: LINE BUFFER OVERFLOW).

```
FOR I% = 1 TO 1000 : A$ =
"CDEFGH" : NEXT I%
```

is sneller dan

```
FOR I% = 1 TO 1000
A$ = "CDEFGH"
NEXT I%
```

Tip-3 (Interpreter)

Zet zo weinig mogelijk blanks " " in een regel met statements.

```
FOR I%=1 TO 1000:A$="CDEF
GH":NEXT I%
```

is sneller dan

```
FOR I% = 1 TO 1000 : A$ =
"CDEFGH" : NEXT I%
```

Tip-4 (Interpreter & Compiler)

Gebruik zoveel mogelijk de juiste soort variabelen. Dit wordt bepaald door een "trailing" character. Gebruik zoveel mogelijk integers bij rekenkundige bewerkingen. Als dat niet kan, gebruik dan zoveel mogelijk single-precision, dat is sneller dan double-precision.

```
% = short integer
& = long integer
! = single precision
# = double precision
$ = string
Default = !

FOR I%=1 TO 1000:A$="CDE
FGH":NEXT I%
```

is sneller dan

```
FOR I=1 TO 1000:A$="CDEFG
H":NEXT I
```

Tip-5 (Interpreter & Compiler)

Ik wil hierbij de aandacht vestigen op het statement SWAP. Het dient voor het verwisselen van de inhoud van twee variabelen. Dit komt bv. van pas bij een zelf geschreven sorteer-routine.

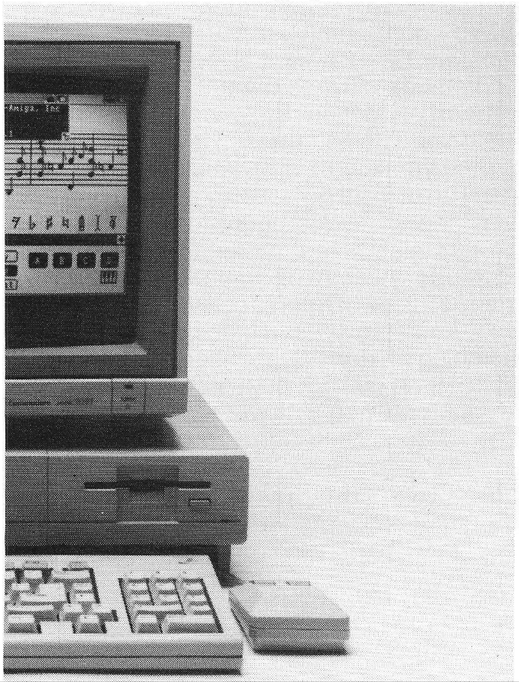
```
SWAP A$,B$
```

is veel sneller dan

```
H$=A$:A$=B$:B$=H$
```

Noot: Als u Tip-1 t/m Tip-3 toepast in de INTERPRETER-omgeving, dan wordt uw programma er niet leesbaarder van, maar WEL veel SNELLER. In de COMPILER-omgeving heeft het geen enkel effect, de compiler doet dit allemaal zelf al. Tip-4, mits consequent toegepast, kan een verrassend groot effect hebben (2x zo snel), maar het effect is sterk afhankelijk van het feitelijke programma. Tip-5 is een losse opmerking.

R.C.A.B. Kohler
Gerrit van der Veenstraat 37
3762 XH Soest



Computer Aided Design en Drafting, kortweg CAD(D), op de Amiga is in feite een prima huwelijk. Aan de ene kant de grafische mogelijkheden van de computer, aan de andere kant de technische tekenopties van de software. Het resultaat kan variëren van logo's of reklameplaten tot technische en bouwkundige tekeningen. Aegis Draw 2000 is zo'n CAD(D)-pakket.

Draw 2000

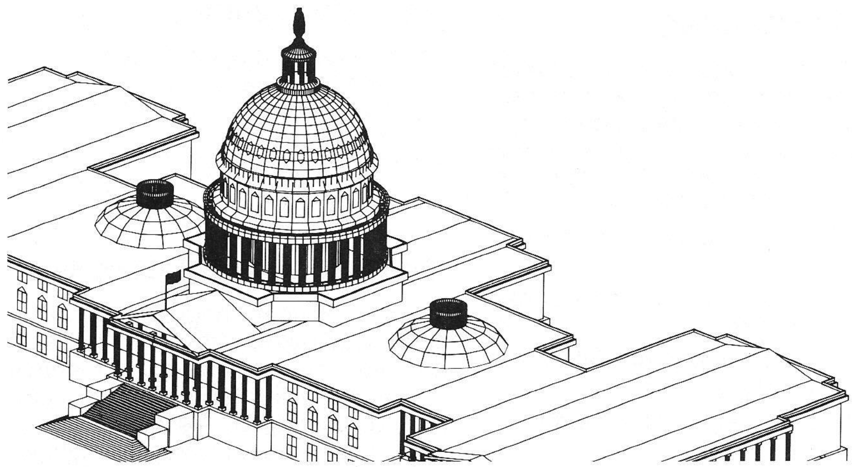
CAD(D) op de Amiga

Professioneel technisch tekenen en ontwerpen is met de grafische kwaliteit van de Amiga goed mogelijk. De enige uitbreidingen die u nodig heeft, zijn een CADD-pakket, een geschikte printer of plotter en liefst meer dan 1 MB aan vrij RAM. Met behulp van Aegis Draw 2000 en een geschikte printer kan de Amiga-gebruiker meteen aan de slag. Bouwtekeningen, industriële ontwerpen, logo's en art work, het behoort allemaal tot de mogelijkheden.

Vroeger was CAD een aangelegenheid voor de professional. Dure mini- of mainframe-hardware en prijzige tekenpakketten maakten CAD onbereikbaar voor de kleinere gebruiker. Gelukkig bracht de komst van desktopcomputers tevens goedkopere CAD-pakketten op het bureau. Kleinere architectenbureau's, technisch- en industriële ontwerpers, vormgevers en reklametekenaars kregen daarmee een krachtig tekeninstrument in handen.
De standaard voor CAD op de PC is **AUTOCAD**. Deze meester onder de CADD-pakketten biedt feitelijk alles wat het technisch tekenhartje begeert. Helaas tegen een prijskaartje van meer dan *f* 6.000,- en daar komt de PC in kleurenuitvoering nog bij. Op de Amiga kan het echter veel goedkoper. In principe kunt u met een Amiga 500, twee diskdrives en 1 MB aan vrij RAM plus een CAD- pakket van 680 piek al aan de slag. Alles uiteraard in de voortreffelijke grafische Amiga-kwaliteit.

Wat is CAD(D)

Onder **Computer Aided Design**, CAD, verstaat men kortweg technisch tekenen en ontwerpen op de computer. Coördinatenstelsels in diverse meeteenheden, hoeken, hellingen, spiegelingen, roteren, rastering, filling, vrij of het automatisch tekenen van standaardelementen (punten, lijnen, bogen, cirkels, ellipsen, vierkanten en veelhoeken), het toevoegen



van teksten of verklarende symbolen en het op schijf bewaren behoren tot de mogelijkheden.
Dikwijls spreekt men van **CADD**. Die extra **D** slaat op drafting, hetgeen in de praktijk zoiets betekent als professioneel tekenen. Hieronder vallen de grafische vormgeving, reclametekenen en artwork.
Op de keper beschouwd is een CAD(D)-pakket niets meer of minder dan een geavanceerde tekenhulp. Het pakket tekent de door de gebruiker opgegeven vormen en figuren. Dat kan met de vrije hand, automatisch of door het inladen van een reeds bestaande tekening. De grote voordelen van CAD(D) t.o.v. normaal tekenen zijn de hoge mate van nauwkeurigheid, de vele bewerkingsmogelijkheden en het gemak van het automatisch tekenen of veranderen van de dimensies. Een perfecte rechte lijn ontstaat door het invoeren van slechts twee punten. Idem dito een curve door het intypen van drie coördinaten. Een vierkant is een kwestie van met de muis de dimensies op het scherm aangeven. Enzovoort.
De gemaakte figuren kunt u weer vergroten/verkleinen, spiegelen, roteren, arceren of in de gewenste kleur uitvullen, gebruiken als bouwstenen voor grotere elementen, arceren, van verklarende teksten of symbolen voorzien, in 2 of 3D-perspectief weergeven, kopiëren en in/uitzoomen. Is de tekening af, dan wordt deze voor later gebruik op de disk opgeslagen. Moet er achteraf nog iets veranderd worden, dan is even ophalen en op de monitor bijwerken voldoende. Vroeger betekende een dergelijke procedure meestal helemaal overtekenen!

CADD wordt voor de volgende doeleinden gebruikt:

- **Architectonische ontwerpen**. Bijvoorbeeld de bouw van een huis of een schuurtje, binnenhuisarchitectuur en stedelijke planning.
- **Werktuigbouwkundige ontwerpen**. Bijvoorbeeld een versnellingsbak, een pomp, een nieuwe motor of de aerodynamische vorm van een vliegtuig.
- **Technisch tekenen**. Met name bij artistieke of grafische vormgeving, het ontwerpen van logo's, desktoppublishing en reclametekenen.



- **Laagsgewijs ontwerpen**: Bijvoorbeeld electronische printplaten en onder elkaar gelegen stelsels van leidingen. In het laatste geval koppelt men vaak een database met gegeven over diepte, lengte en breedte, aard van, ouderdom en onderhoudstatus aan de tekenlagen.
- **Simulaties**. Beproeving van de ontwerpen door het aanbrengen van druk-, trek- of rotatiekrachten, hitte of koude en versnellingen. Deze simulaties kunnen veel geld besparen, leveren geen gevaar op en kunnen zelfs in andere proefopstellingen niet haalbare extreme condities nabootsen.
- **Koppeling aan een productieproces**. Bijvoorbeeld een computerbestuurde freesbank. Men spreekt van **CAM**, computer aided manufacturing.
- Een wat oneigenlijk CAD(D)-gebruik is het ontwerpen van **(organisatie-)schema's**, **kaartjes** en **technische handboeken**.

Natuurlijk zijn er qua gebruikers grote verschillen. De een zal een CAD(D)-pakket voor industriële vormgeving gebruiken, een ander bouwt er een schuurtje mee en de volgende wil er zijn teksten mee verfraaien. Gaat het u daarentegen alleen om wat kleurige schetsjes op het scherm, dan is het goedkoper om een gewoon paintpakket zoals DeLuxe Paint II aan te schaffen. Dat werk nog makkelijker ook.

Wat is nodig voor CAD?

De minimum-configuratie voor bureau-CAD(D) verschilt sterk per gebruiker. Er is bijvoorbeeld een heel verschil tussen een tekening voor het ontwerp van een hydraulische pomp en een illustratie voor desktoppublishing. Nietemin zijn er altijd dezelfde basiscomponenten nodig. Centraal staat de **PC** in dit geval een Amiga 500, 1000 of 2000. Voor het zwaardere werk raden we een model 2000 aan. De standaard Commodore 1081 **kleurenmonitor** is voor doorsnee CAD-werk net voldoende. Een professional ambieert waarschijnlijk een monitor met een hoger oplossend vermogen en de bijbehorende extra videohardware.

Gedetailleerde graphics vreten zoals bekend RAM- en schijfruimte. Zorg daarom voor minimaal 1 MB (liefst 2-4 MB) aan vrij **RAM** in de Amiga en minimaal **twee drives** waarvan bij voorkeur één een **harddisk** is. De **printer** of **plotter** bepalen de kwaliteit van de uitvoer. Met name bij kleur en de wat grotere tekeningen verdient een compatibele plotter de voorkeur. Een kleuren-matrixprinter kan in deze een goedkoper alternatief blijken. Bij zwartwit voldoet een laserprinter met de juiste drivers prima.
Verder is er nog een **invoerdevice** nodig. Behalve tekenen met de muis is er keuze uit trackballs, lichtpennen, digitizer tablets en pucks. De professional prefereert de dure puck met bijbehorend digitizer tablet. Met het toetsenbord alleen zijn niet alle tekenopties bruikbaar! Bovendien werkt het keyboard in vele gevallen onhandig.
Voor supersnelle rekenkracht komt tenslotte een Motorola 68020 versnellerkaart of mathematische 68881 coprocessor in aanmerking.

Draw 2000

De toevoeging 2000 suggereert dat er een Amiga 2000 nodig zou zijn. Dat is gelukkig niet zo, hoewel de software van minimaal 1 MB aan vrij RAM en twee diskdrives uitgaat. Voor de goede orde probeerden we **Aegis Draw 2000** ook op een oude Amiga 1000 met 512 KB aan boord hetgeen voor eenvoudig tekenwerk redelijk ging. Bij gedetailleerde en grote tekeningen blijkt die 512 KB echter al snel te krab met alle noodlottige gevolgen van dien. Houdt u daarom aan de pakketspecificaties.
Volgens de inleiding van Aegis biedt Draw 2000 zo ongeveer het AutoCAD-mekka op de Amiga. Dat is natuurlijk lichtelijk overdreven, hoewel de software in onze praktijktest hoog scoorde. Voor kleinere industriële of technische ontwerpen, architectuur en met name studie van CAD(D), desktoppublishing (IFF-files) en tekendoeleinden is het pakket heel geschikt. Bovendien maakt de gunstige prijs veel van de missende AutoCAD-opties (die de kleine gebruiker toch niet nodig heeft) goed.
Aegis Draw 2000 werkt in principe net als elk ander CAD(D)-pakket. De software verschaft de gebruiker een **venster** waardoor hij/zij naar de tekening kijkt. Het venster laat slechts een deel (in detail) van de grotere tekening zien en heeft een lagere resolutie dan het uiteindelijke printer/plotter-resultaat. De mate van zoom-in bepaalt hoe gedetailleerd u de tekening be-

kijkt. Binnen het geopende venster wordt het desbetreffende deel van de tekening bewerkt. Met 1 MB of meer aan vrij RAM in (of aan) de Amiga is het mogelijk om tegelijkertijd met meerdere geopende vensters te werken. Die geopende vensters kunnen dan delen van één tekening bevatten of meerdere tekeningen die later tot één geheel gekombineerd zullen worden. Centraal in het CAD(D)-gebeuren staat het **design-proces**. Onder design wordt het feitelijke tekenwerk verstaan. Dat kan bestaan uit:

- Het **inladen van objecten** uit de database met tekenelementen en deze objecten verder bewerken.
- Het **automatisch laten tekenen** van figuren volgens de opgegeven specificaties plus de eventuele verdere bewerkingen.
- Het **vrij tekenen**, d.w.z. uit de losse muispols tekenen.
- Een **combinatie** van de drie hiervoor vermelde mogelijkheden.

Een groot verschil met een conventioneel paintpakket is dat Draw 2000 de tekening, **drawing**, uitsluitend een **representatie van de feitelijke tekening** laat zien. De oorspronkelijk graphicsplaat blijft met alle bijbehorende specificaties gestructureerd in het in het RAM-geheugen zitten. Bij een wijziging of bewerking verandert slechts de display voor de desbetreffende verandering(en). Dat maakt zeer gedetailleerd inzoomen, bewerking in vensters, elders onder gewijzigde specificaties optekenen, afdrukken onder een hogere resolutie dan die van de display, laagsgewijze-opbouw van de tekening en ingewikkelde bewerkingen mogelijk.

De praktijk

Aegis Draw 2000 is een **gereedschap-georiënteerd** CAD(D)-pakket. D.w.z. de gebruiker tekent met de (muis-)hand op het monitorscherm en pakt daarvoor één van de aanwezige tools. Een tool kan bijvoorbeeld zijn een tekenpen (in verschillende diktes), een vlakgummetje een automatisch tekensymbool (cirkel, vierkant), in patroon-invultool of een resizer. Het **toolmenu** bestaat uit twee kolommen: de linker kolom bevat de ontwerpgereedschappen en de rechter kolom bevat de bewerkingstools. Aparte menu's dragen zorg voor de invulling van arceringen en kleuren.
Draw 2000 maakt onderscheid tussen **tijdelijke** en permanente tools. Tijdelijke gereedschappen (tempory tools) zijn bijvoorbeeld een ERASE en een ZOOM IN. Het tijdelijke aspect blijkt uit het feit dat Draw, na beëindiging van de tooltaak, direkt weer naar het gereedschap waarmee de gebruiker eerst bezig was, terugkeert.
Zoals gebruikelijk bij de Aegis Amiga-programma's maakt Draw 2000 weer volledig gebruik van de werkbank. De menustructuur en muisklik-activatie zijn identiek en het CAD(D) werkt dan ook volledig menugestuurd. Het hoofdmenu biedt de keuze uit de opties:



- **Project** voor het maken van nieuwe tekeningen, het inladen en saven van grafische (IFF-)bestanden of delen daarvan, kleuren en standaardinstellingen, het wissen van schermtekeningen en het afdrukken op printer of plotter.
- De **Edit-optie** dient zoals de naam al aangeeft voor het bewerken van het tekeningdeel binnen één (= het actieve window) van de geopende vensters. Bijvoorbeeld: CUT en PASTE, COPY, ROTATE, RESIZE, MIRROR en UNDO. U kunt gehele objecten of delen daarvan bewerken.
- Het **Tools-submenu** voor het creëren van de figuren en vormen.
- Onder **Display** staan de commando's voor het bekijken van het actieve venster. Bijvoorbeeld wel of geen linealen in beeld, de achtergrondkleur, het gebruik van rasters, welke laag wordt bekeken, de werkresolutie (640 x 400 of 640 x 200 pixels) en het zoomniveau.
- De **Options** bevat opdrachten voor o.a. de lijndikte, afmetingen van de rasters, de fontkeuze, lijnpatronen, de mate van tekennauwkeurigheid en plotterschaal.
- Onder de **Preferences** staan de voorkeursinstellingen naar wens op aan of uit.
- Het **Color-menu** beheerst het bruikbare kleurenpalet.

Er is ook een zogenaamd **Fast Menu** dat alle tools, het kleurenpalet, de zoomopties en Undo in een Amiga-venster onderbrengt.
De **vensters** kunnen op eenvoudige naar keuze vergroot/verkleind (resizing) en naar een andere schermpositie verplaatst worden. Verder kunt u de windows desgewenst openen/sluiten en er mee door de te bekijken tekening scrollen.
Het Engelse hanboek is duidelijk en volledig en zal voor de Amiga-gebruiker die iets over computertekenen weet een flink eind op weg helpen. De finishing touch komt pas na lang oefenen. Oefening baart immers kunst.

Overige specificaties

De met Aegis Draw gemaakte tekeningen kunnen in andere **IFF**- compatibele pakketten gebruikt worden. Bijvoorbeeld in Aegis Modeller 3D, Lights!, Camera! Action!, Video Titler (logo's), Amiga dtp-software en DeLuxe Video.
Een **genlock** opent geheel andere interessante perspectieven. Tekeningen, kaartjes, schema's en logo's kunnen rechtstreeks met bestaande videobeelden gecombineerd opgenomen worden. Ook is het mogelijk om bestaande tekeningen met behulp van een videocamera en genlock binnen een Draw 2000-venster te halen en daar verder te bewerken.
Bij het zwaardere werk komt een 68020 versnellerkaart of een Motorola 68881 mathematische coprocessor goed van pas. De snelheidswinst betaalt die extra investering dubbel en dwars terug.

Aegis Draw 2000 biedt behoorlijk wat waar voor zijn geld. AutoCAD wordt echter niet geëvenaard en dat is ook niet nodig voor de beoogde doelgroep. Met name de kleinere ontwerper, student en grafisch vormgever kunnen met dit CAD(D)pakket hun voordeel doen. Draw 2000 kost inclusief de coprocessorversie en handboek (totaal 3 schijven) *f* 679,-.

Inlichtingen: Altycos Imports, Zoetermeer, tel.: 079-510757.

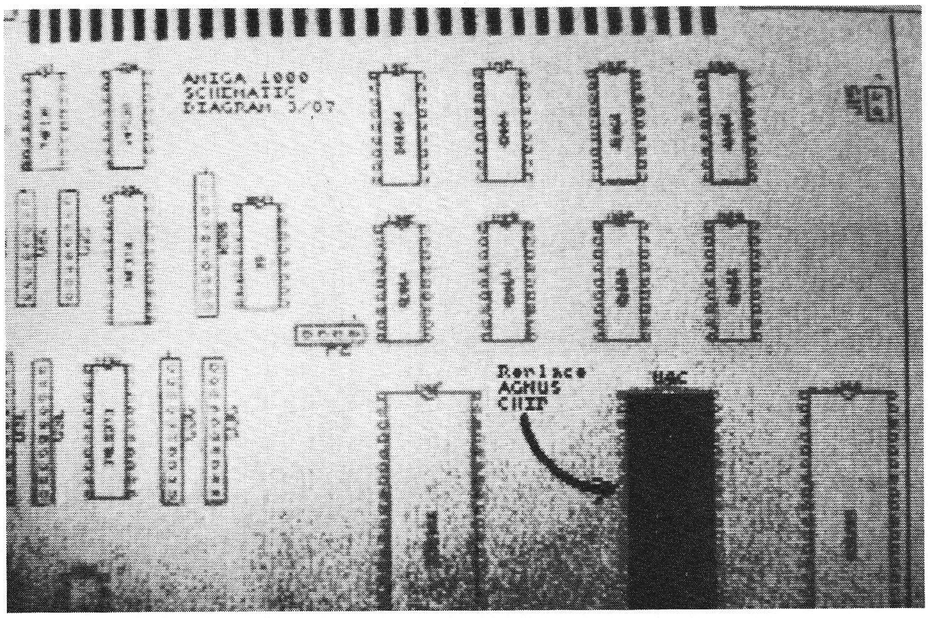
Een naam als Magellan doet aan een adventure over deze oude ontdekkingsreiziger denken. De digitale Magellaan blijkt echter een ontwikkelsysteem voor expertsystemen te zijn. Een kunstmatig intelligent pakket dat bedoeld is om de beslissingsregels en kennis van de menselijke expert in een computerprogramma vast te leggen. Volgens de maker van Emerald Intelligence is Magellan in staat om over letterlijk elk onderwerp, van diagnostic tot decision support, een expertsysteem samen te stellen.

Magellan

Een expertsysteembouwer voor de Amiga

Het AI-pakket Magellan versie 1.1 beoogt het op eenvoudige wijze, d.w.z. zonder kennis van van programmeertalen zoals LISP of PROLOG, construeren van een expertsysteem. De menselijke expert kan zijn/haar kostbare kennis en ervaring zonder problemen op de ontwerpschermen (= box-filling) invoeren. Daarbij maakt Magellaan aanvullend gebruik van eenvoudige IF-THEN-uitdrukkingen, graphics, animaties en mathematische berekeningen. Verder kan de ontwerper een eenmaal gemaakte kennisdatabase gemakkelijk met extra informatie uitbreiden.

In principe is **Magellan** een venster/menu/aanwijzer-interfaceprodukt, kortweg een system building tool, voor expertsystemen. Een **expertsysteem** is een kunstmatig intelligent stuk software dat de menselijke logica, ervaring en intuitie op een bepaald kennisgebied nadoet. Bijvoorbeeld de kennis van een meester-automonteur in een diagnosesysteem voor motorische problemen van de autotypen XYZ. Expertsystemen kunnen de menselijke expert tot op zekere hoogte vervangen. Overal in het veld dure experts stationeren voor een weinig gebruikt of zeer ingewikkeld vakgebied heeft weinig zin. In zo'n geval kunnen een kennissysteem op de PC en een vragen intoetsende geschoolde niet-expert de meeste zaken zelf best af. De software stelt de operateur, jurist, arts, technicus, monteur e.d. een aantal relevante vragen en beredeneert volgens de ingebouwde beslissingsregels de juiste diagnose, remedie of behandeling.
Met name bij de eenvoudige, maar lastig te vinden problemen, ingewikkelde behandelings- of reparatiemethoden, juridische adviezen en decision support (= beslissingsondersteuning) kan een expertsysteem goudwaard blijken. Alleen bij de echt precaire of zeer moeilijk te doorgronden gevallen is de aanwezigheid van de echte menselijke expert op de werkplek nog gewenst.



Magellan

Het expertsysteem

Een doorsnee expertsysteem bestaat uit drie belangrijke componenten:

- De kennis-database of **knowledge-base**. Daar in staan alle voor de te nemen beslissingen nodige vragen, kennis, waarschijnlijkheden, ervaringsfeiten, plattegronden, electronische circuits, mechanische modellen, behandelingsschema's enz.
- Een beslissingsmodule, de **inference engine**, voor het maken van logische deducties, het trekken van conclussies en het geven van adviezen.
- Een gebruikersshell, de **user-interface**, voor de gebruiksvriendelijke communicatie met de gebruiker en het vastleggen van de kennis en beslissingsregels.

Magellan wordt geheel maagdelijk bij u afgeleverd. De potentiële computer-

expert weet dus nog helemaal niets. Alle expertise dient nog door de kennisingenieur te worden ingevoerd. Dankzij de relatieve eenvoud van de bediening en de wijze van systeemontwerp kan elke expert of toekomstige gebruiker de benodigde kennis, regels en feiten zelf invoeren.

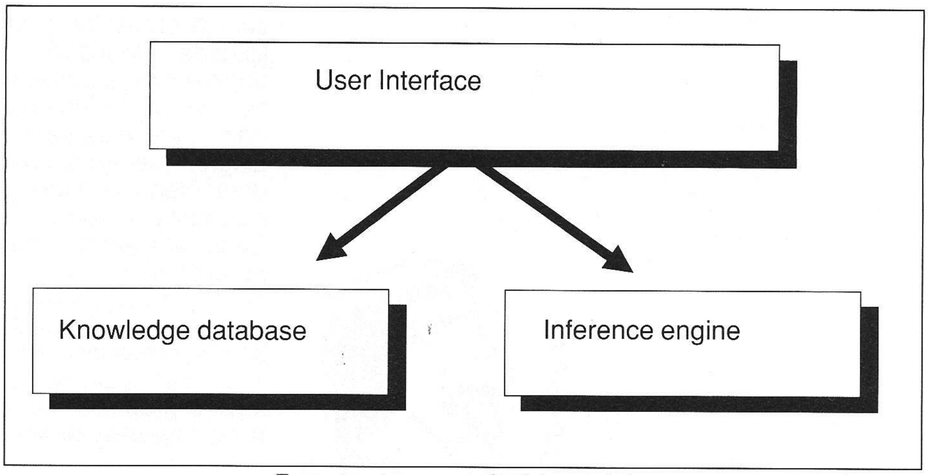
De inference engine

De "gevolgtrekkingsmachine" van Magellan beschikt over alle gereedschappen om een goede digitale beslisser te ontwerpen. In de meeste gevallen zal de logische deductie bij de vraagstelling van de gebruiker beginnen. Wat wil de vraagsteller weten? Zodra de database de vraagstelling onderkend heeft gaat de inference engine aan de slag. De knowledgedatabase van de expert- beslissingsregels wordt gesorteerd op die regels die kunnen helpen bij het vinden van de juiste oplossing of advies.
Er zijn twee belangrijke redeneerprocessen:

- Het volgens de ingetypte gegevens naar een bepaalde conclusie toe redeneren. Men spreekt van **foreward chaining**. Deze techniek wordt vooral gebruikt bij (productie-)planning en responseexpertsystemen. Daarbij staan tijdsafhankelijke beslissingsregels centraal.
- Een bepaalde conclusie wordt op zijn mogelijke waarheid getoetst. Men spreekt van **backward chaining**. Een techniek die veel gebruikt wordt bij problem solving, diagnostische en andere doelgerichte expertsystemen.

De meeste beslissingsregels verlopen via de IF-THEN -structuur. Indien A dan geldt B. Magellaan vergemakkelijkt de redeneerkeuze door deze van de gebruiker over te nemen. In praktijk zal de software zowel back als foreward chaining proberen en zoveel mogelijk conclusies trachten te trekken. Het enige wat de gebruiker behoeft te doen is het beantwoorden van de gestelde vragen.

Verder behoren o.a. tot de inference engine-tools en opties: (desgewenst automatische) zekerheidsfactoren met drempelwaarden, zowel IF-THEN- als JA-NEE-regels, een synergistische inferencemodus, bidirectionele regels, parsing van algebraische uitdrukkingen, ondersteunende informatie-, grafische (IFF)- en animatiefiles, het WHY? uitleg-utility, legale waarden en beperkingen, cellen met



Expertsysteem met de drie modulen

meerdere waarden en antwoorden, context-sensitieve hulp, een door de gebruiker te definieren dictionaire met synoniemen, problemen opslaan (ook met snapshot) en weer ophalen. De voertaal van de inference engine, interface en database is het Engels.

De knowlegde-bases

Het aanleggen van **kennisdatabases** met Magellan kan van de grond af aan via de data entry-schermen of met behulp van al bestaande databases. Magellan beschikt standaard over het Interface Group Module dat de meeste voor de Amiga gangbare database- en spreadsheetformats kan inlezen. Een tweede importeeroptie biedt de AREXX-compatibiliteit voor communicatie via de AREXX-standaard. Verder kan het programma een aantal C-broncodes inlezen.
Met het **AREXX-interface** zijn interessante dingen mogelijk. Wat dacht u bijvoorbeeld over het kunstmatig intelligent componeren van muziek via een MIDI-editor? Leer Magellan de muziekregels en er zit een huis-tuin-en-keuken-componist in de Amiga. In de praktijk kan Magellan andere Amiga-programma's als een soort processes runnen. Bijvoorbeeld de besturing van graphics en animaties.
De **technische specificaties** van knowledge-base zijn:

- Memory residente database met een maximale omvang die gelijk is aan het systeem-RAM
- Disk-swapping tot aan de schijflimiet. Bij harde schijven zijn dus zeer grote databases mogelijk
- Meer dan 500 beslissingsregels in een 1 MB Amiga. 50- 100 regels in 512 KB
- File-compressie voor efficientere opslag en inladen
- Optimale (onder C met assembler) sortering, toegang en selectie van regels en datacellen. In C zou Magellan 3-8 maal sneller dan onder LISP draaien

Bij grotere kennissystemen zijn RAM-uitbreiding en een hard disk eigenlijk onmisbaar.

De user-interface

De **gebruikersinterface** moet Magellan openleggen voor de niet-programmeur met veel kennis, maar weinig inzicht in het ontwerpen van expertsystemen. Gebruikmakend van de visuele oriëntatiemogelijkheden op de Amiga krijgt de ontwerper overzichtelijke invulboxes voorgeschoteld. Het opstellen van regels en het invoeren van de databasekennis wordt daarmee aanzienlijk vereenvoudigd. De input bestaat uit Engelse tekst, wiskundige expressies (waaronder tal van variabelen) en menu-opdrachten. Bedenk wel dat u ook niet met Magellan aan het uitgebreide voorbereidingswerk komt. De kennis van de expert dient eerst in letters, figuren en beelden te worden ondergebracht alvorens deze data ingetypt of gescand kunnen worden! Hoe beter en overzichtelijker de voorbereiding des te gemakkelijker kunt u de vragen en verlangens van deze expertsysteembouwer vervullen. Dat bespaart veel tijd en ergernis.

Alle begin is moeilijk en het blijkt verstandig om eerst eens met de box-filling te experimenteren. Lukt het om de vraag- en regelboxes goed in te vullen, dan wordt het tijd om eens gebruik te gaan maken van de geavan-



ceerde grafische, animatie-, geluids- en mathematische mogelijkheden. Magellan maakt gemakkelijk interface-kontakt met andere Amiga-software. Bijvoorbeeld grafische, muziek, animatie-, calculatie- en dbms-pakketten. Daarmee kunt u allerlei extra's zoals werktekeningen, plattegronden, simulaties, stroomdiagrammen (o.a. charts voor projectmanagement), geluidsfragementen (van storingen) en behandelingsschema's in de kennisdatabase laden.
Centraal staat het **Window/Menu/Mouse-interface** dat het opbouwen van de beslissingsregels en de invoer van data in de cellen verzorgt. Daarnaast beschikt Magellan over bit-mapped graphics en animaties, command file execution met I/O-redirection, synthetische spraak en geluid, seriële en parallele data-interfaces, legal value "buttons", printouts en de combinatie van tekst & graphics. Kortom al het gereedsschap om een **interactieve expertsysteem** van de grond af te kunnen opbouwen.

Magellan is een voorbeeld van de nieuwe generatie expertsysteem-bouwers. Overal waar een interactief expertsysteem voor het oplossen van de dagelijkse problemen gewenst is zal dit Amiga- pakket goed voldoen. Het construeren van een kunstmatig intelligent expertsysteem valt met behulp van het gebruikers- interface relatief eenvoudig te noemen. Wel zijn enige achtergrondkennis en het nodige voorbereidende werk vereist. Verder mag u natuurlijk geen wonderen van de gemaakte expertsystemen verwachten. Bij de extremere gevallen zal de menselijke expert er weer aan te pas moeten komen.
Magellan is een produkt van Emerald Intelligence en wordt in de PAL-versie halverwege 1989 verwacht. Het pakket is voorzien van een uitgebreid Engelstalig handboek en een aantal demodatabases. Optioneel is een eigen Magellan-Nieuwsbrief verkrijgbaar. Een exacte prijs en importeur waren op het moment van schrijven nog niet bekend.

U.S

Met een genlock-interface voor de Amiga is het mogelijk om grafische computerbeelden en titels met videopnamen te combineren. Genlocks zijn tegenwoordig in alle prijs- en de daaraan verbonden prijsklassen verkrijgbaar. Professionele genlock-interfaces kosten rond de *f* 4.000,- en dat is te duur voor de gemiddelde amateur. Een voor de thuis-videostudie goed functionerende Amiga-genlock kost net iets minder dan 1.000 gulden. We bekeken in deze categorie de Rendale A8802 voor u.

Rendale A8802 Genlock

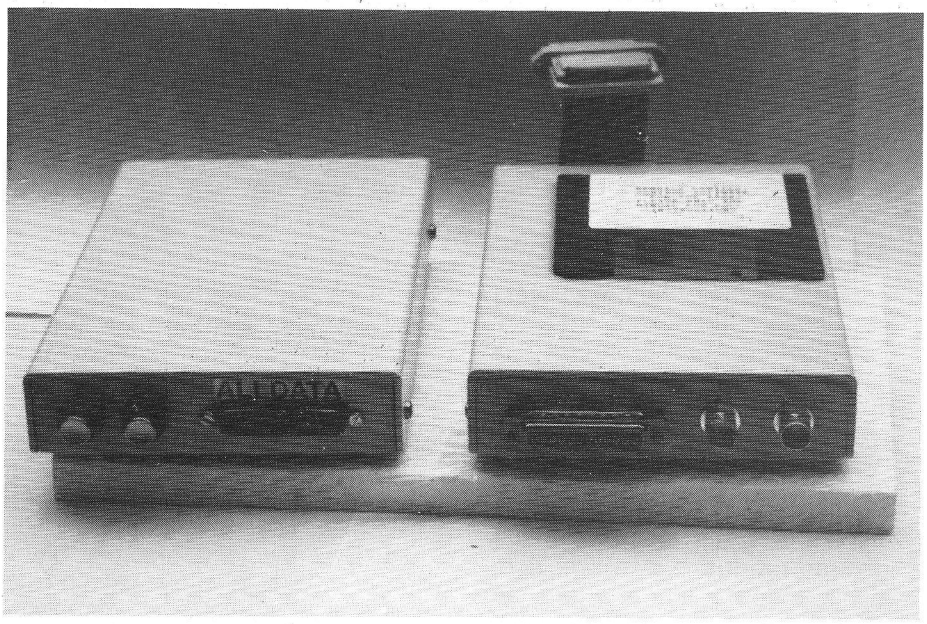
Budget-prijs genlocking op de Amiga

De Amiga 500, 1000 en 2000 staan al jaren lang in het middelpunt van de groeiende belangstelling voor desktop video (DTV). De grafische prestaties en animaties van deze machines zijn beroemd en vragen gewoon om combinatie met life opgenomen videobeelden tot een flitsende moderne productie. Redelijk geprijsde PAL videokaarten en genlocks brengen tegenwoordig de 'videostudio op het bureau' binnen het bereik van de Amiga-hobbyist en kleine videoproducent.

Genlockers combineren zoals gezegd computervideo- met bestaande of life opgenomen videobeelden. Men spreekt ook wel van beeldmixen. Bekend zijn de titelletters die over de videobeelden heenlopen, omroeplogo's en de combinatie van animatiefiguren met echte spelers. Enkele moderne videocamera's beschikken al standaard over een **beeldgeheugen** of ingebouwde **titelgenerator** en maken on the spot titels of voegen deze achteraf nog toe. De mogelijkheden van dit soort cameratrucjes zijn echter beperkt in vergelijking met wat een **videocomputer** zoals de commodore Amiga allemaal kan.

Het gebruik van een computer als aftitelaar en grafische effectgenerator biedt dus veel meer mogelijkheden dan een gewone ingebouwde of losse titelgenerator voor video. Niet alleen kan de videomaker kiezen uit **meerdere fonts** (o.a. van Zuma) in vrijwel elke bruikbare grootte en kleur, ook kan men de titels laten scrollen, roteren om een X-, Y- of Z-as, van schaduw- of stroboscoop-effecten voorzien, creatief de belichting (lichtval) besturen en van vorm laten veranderen. Professionele **titelsoftware** zoals bijvoorbeeld TV*TEXT weet wel raad met dergelijke effecten en in een mum van tijd beschikt u met de Amiga over een echte titelstudio.

Behalve titels kunnen ook grafieken, geanimeerde poppetjes of logo's, grafische figuren, artwork en met paintpakketten ontworpen achtergronden met life/bestaande videobeelden gecombineerd worden. Hiervoor kunt u een conventioneel **tekenpakket** zoals DeLuxe Paint III of speciale **videosoftware**, bijvoorbeeld DeLuxe Video en Aegis Animator, gebruiken. Reclamebureau's, kabelkranten, omroepen, filmmakers en videohobbyisten passen dit soort videocomputertechnieken in combinatie met een genlock-interface steeds meer toe.



De A8802 Genlock

Hoe werkt een Genlock?

Een genlock heeft doorgaans twee video-ingangen en één of meer video-uitgangen. Op de video-in worden twee verschillende videobronnen aangesloten. Bij de **A8802** zijn dat de Amiga via een korte 23-polige RGB D-connector en een composiet-video-ingang (CVBS BNC-in) voor een videocamera, videorecorder, beeldplaat, een andere computer, een satellietsignaal of een professionele "black and burst"-input. De video-timing-signalen voor line, frame en co-

lour burst van de Amiga en de externe videobron worden door het interface **gelocked**. Dat wil zeggen dat de Amiga en externe videobron zodanig op elkaar afgestemd worden, dat beide signalen dezelfde kleur op dezelfde tijd en plaats op het scherm (of naar de videorecorder) geven. Men spreekt hierbij ook wel van synchroniseren van twee videobronnen. Zijn de beide videosignalen eenmaal gelocked, dan kan de electronica tussen het Amiga- en externe videosignaal switchen. Daarvoor gebruikt de genlocker een high speed videoschakelaar. Met behulp van deze schakelaar is het mogelijk om videobeelden als het ware over elkaar heen (**overlay**) te leggen, waardoor gemixde videobeelden ontstaan.
Behalve het eigenlijke locken vervult de Amiga genlock A8802 nog twee andere belangrijke taken:

- **(graphics encoding**, het omzetten van het Amiga **RGB**- signaal in een voor de monitor en/of videorecorder bruikbaar **composiet PAL- signaal**. Voor het functioneren van de encoder is een PAL-referentiesignaal nodig om de computersignalen opnieuw te timen.
- **Video overlay switching**, het schakelen tussen het computer- en het externe videosignaal. Er zijn vier schakelmogelijkheden: Amiga Display, display externe videobron (default), de background- en foreground-modus. In de **background-modus** wordt de achtergrondkleur doorzichtig (transparant) en het externe videobeeld (CVBS-signaal) zichtbaar in mixage met de computer graphics, bijvoorbeeld een aftiteling. Het register 0 wordt hier als de schakelaar gebruikt. In de **foreground-modus** is zwart de doorzichtige kleur. Daarmee kan een (voor- en achtergrond-)kleur transparant gemaakt worden om een extern videobeeld in een venster te mixen.

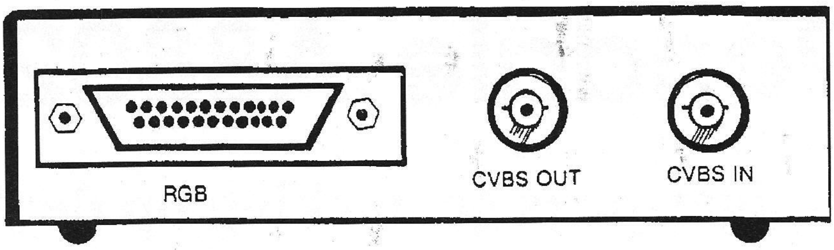
Voor de expert is er de mogelijkheid om Black and Burst, het aanbieden van een effen (zwarte) achtergrond voor overlays te gebruiken. De amateur zal in de meeste gevallen een normaal composiet videosignaal voor de overlay gebruiken.

De A8802

De door Cat & Korsh en Alldata geïmporteerde **A8802 genlock** is grotendeels gelijk aan de wat oudere 8702-versie. De oude versie was te koop met en zonder twee drukknoppen voor het switchen van de verschillen videomodi. Op de door ons geteste Rendale 8802 werd dit schakelen echter softwarematige van uit de Amiga gestuurd. Na installatie van de software kunt u de verschillende videomodes met de eerste vier functietoetsen of de cijfers 1 t/m 4 besturen Uiteraard beschikt de A8802 over alle hiervoor genoemde video-mogelijkheden en heeft drie verschillende aansluitingen:

- Een gewone **composiet BNC**-uitgang voor het aansluiten van een videorecorder of monitor. Uit deze output komt het gecombineerde (overlay) PAL-signaal.
- Een **BNC CVBS video-ingang** voor de video-input uit kamera, VCR of beeldplaatspeler
- De **23-polig RGB Amiga D-connector** voor: Rechtstreekse aansluiting van de eigen Amiga-monitor en -videohardware.

Het genlock-interface is een net plat metalen kastje ter grootte van een pocketboek. Uit dit kastje komt een lintkabeltje voor aansluiting op de Amiga D-connector. Dit kabeltje bleek voor onze Amiga 1000 wat aan de korte kant om het interface gemakkelijk bereikbaar naast de machine te zetten. Gelukkig is de besturing geheel softwarematig, zodat je niet steeds aan moeilijk bereikbare knoppen behoeft te wriemelen. Lastig is wel het los- en vastmaken van de BNC pluggen als de genlock achter de Amiga zit.
Het aansluiten van de genlock geeft geen probleem, zolang men de nodige verloopjes in huis heeft. Rendale levert de korte lintkabel er natuurlijk bij. Deze is van een afwijkend TTL-type. Zelf er eentje solderen kan tot het opblazen van de RGB-poort leiden. Voor de videokabeltjes moet u zelf zorgen. Videorecorders gebruiken tulp-, SCART- en BNC-pluggen. Voor tulp en Scart zijn verloopstekkers en kabels bij elke goede video- of electronicazaak te koop. Neem wel enkele tientjes mee want goedkoop zijn deze supplies beslist niet!

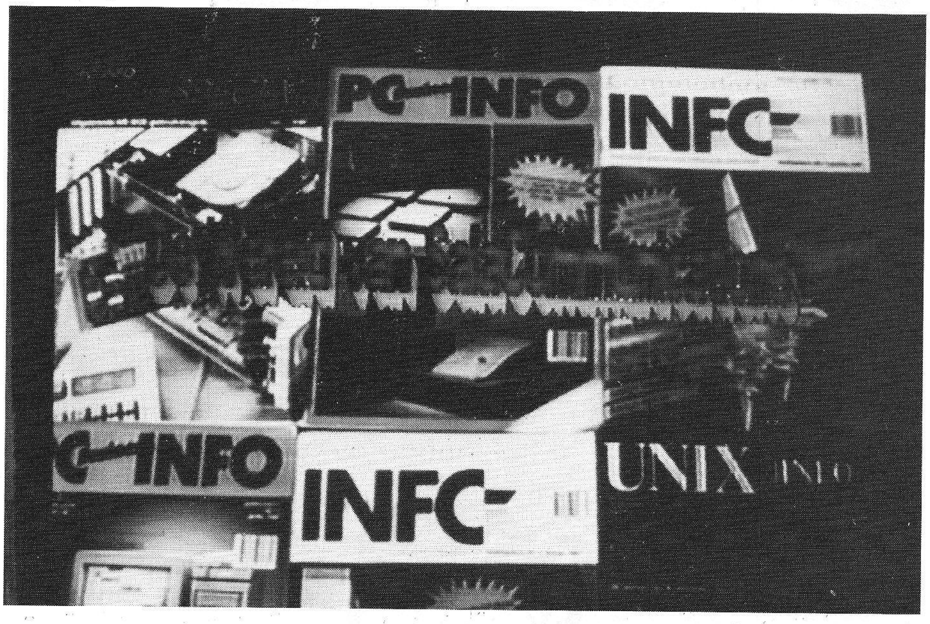


Aansluitingen achterzijde Rendal A8802 genlock

De videopraktijk

Eerlijk gezegd zijn we altijd een beetje sceptisch ten opzichte van amateurgenlocks. Professionele videokwaliteit kost geld en de amateur-hardware is relatief gezien goedkoop. Daar kwam nog bij dat de eerste amateurgenlocks een ronduit slechte beeld- en opnamekwaliteit afgaven.
Dankzij de sterk groeiende DTV-populariteit van de Commodore Amiga's en de electronische ontwikkelingen zijn de gelock-prijzen flink gedaald. Topmodellen gaan voor *f* 2.000,- tot *f* 4.000,- over de toonbank. Een redelijk tot goed amateurmodel kost iets van 1.000 piek. Dat laatste gaat ook op voor de Rendale A8802 Genlock. Al bij de eerste tests bleek het allemaal best mee te vallen. Zeker als je van een kwalitatief goede (Super) VHS videorecorder en -camera gebruikt maakt. Rest nog een kwaliteitstape en de gegenlocked opnamen zien er best aardig uit. Het is allemaal nog niet op en top broadcasting- kwaliteit, maar de A8802 kan aardig op het semiprofessionele front meekomen. Niet alleen op de monitor, maar ook op de videorecorder.
Het maken van overlays en het switchen tussen de verschillende modes kan iedereen in vijf minuten leren. Genlocker aansluiten op de Amiga, externe videobron en output, software (kickstart en video- of paintpakket) in de Amiga, de besturingssoftware laden en de videoproductie in de bureaustudio kan beginnen. Grote creatieve mogelijkheden bij een goede videokwaliteit. N.B.: de RGB-kwaliteit van de Amiga- graphics blijft door een grotere bandbreedte en andere kleur-

codering altijd superieur ten opzichte van registratie in composiet-video. M.a.w., de op videotape gerigistreerde Amiga- graphics zullen altijd wat tegenvallen. Experimenteer met verschillende achtergronden en kleurcombinaties voor het verkrijgen van de beste resultaten.
Het gebruik van een **enhancer** voor het wegwerken van de onvermijdelij-



Een toepassing van de A8802 Genlock

ke kopieerverliezen kan soms noodzakelijk blijken bij een wat lagere VHS-kwaliteit. Bij SVHS vallen, ondanks het feit dat de genlock de SVHS-aansluitingen ontbeert, de kopieerverliezen erg mee.
Na het opstarten zijn de default videobeelden op de achtergrond en computer-graphics op de voorgrond, de **background-modus**. De kickstart-ikoon wandelt eerst door het beeld heen. Dat verdwijnt na het inladen van de werkbank of een video/paintpakket. Voor het veranderen van de videomodus gebruikt de videoproducent de meegeleverde besturingssoftware.
Het effect van de fore- en background-modus is afhankelijk van de gebruikte kleurinstellingen. In de background-modus krijgt de gebruiker de beste resultaten als er ook een zwarte achtergrondkleur gekozen wordt. Dat voorkomt het doorlopen van de kleuren. In de **foreground-modus**, waar de Amiga-graphics transparant zijn, verdient het aanbeveling om de preferences naar smaak in te stellen. Aanbevolen wordt om bij de Preferences Rood en Groen minder dan 50% in te stellen en blauw op full te zetten. Hierdoor ontstaat een professioneel blauw overlay-venster. Gebruik nimmer dezelfde voor- en achtergrondkleur, want dan wordt de achtergrond niet doorzichtig. Register 0 is dan inactief. Verder kent de A8802 nog de **Amiga-modus** (alleen de Amiga-graphics zijn zichtbaar) en de **video-modus** (allen de CVBS video-input is zichtbaar). De resolutie bedraagt naar keuze medium in 32 kleuren of high bij 16 kleuren. Let op dat de Genlock een interlace-mode van de Amiga verwacht, anders werkt het interface niet.
Een klein probleempje geeft het encoderen van het Amiga RGB-signaal. Zonder een extern referentiesignaal komt er niet veel bijzonders uit de video-out. De gebruiker zal de genlocker dus een tuner- of ander geschikt videosignaal (kamera, VCR) moeten aanbieden voor het in fraaie PAL-kleuren vertonen van de Amiga-graphics. Dan behoeft de A8802 niet "droog te zwemmen". Een alternatieve mogelijkheid is het aanbieden van een zwart-wit-signaal voor het creëren van zwart-wit-overlays.

Opstelling

De video-configuratie is uiteraard afhankelijk van het beoogde doel. Videomakers zullen de Amiga met genlocker gebruiken voor mixages en afviteling. Daarvoor zijn nodig een videocamera of videorecorder voor weergave, een opnamerecorder en/of output-monitor. Bij voorkeur maakt men gebruik van meerdere monitoren: een voor de Amiga zelf en een voor het controleren van het composiet-outputsignaal. Let er bij het gebruik van een monitor op dat deze op AV/VCR staat, anders wordt de output niet correct weergegeven.
Bij interactief onderwijs maakt men gebruik van een door de Amiga gestuurde beeldplaat- of videorecorder. Het interactieve Amiga- programma stelt de vragen of verschaft de begeleidende teksten. De externe videobron levert de bijbehorende beelden.

Gebruiksmogelijkheden

De toepassingen van de Amiga met genlock zijn legio. Behalve videoproducties, reclamepresentaties, slideshows en interactief onderwijs zijn er nog tal van andere gebruiksmogelijkheden. Alldata noemt zelf trainingssystemen, het gelijktijdig bekijken van twee videobronnen (bijvoorbeeld een computer- en een TV-programma), informatie-terminal en toepassing als een video- database.

Conclusies en opmerkingen

De A8802 Genlock is een welkome aanvulling voor de videoproducent, DTV-hobbyist of onderwijskundige die met de Amiga werkt. Voor *f* 999,- krijgt de koper een kwalitatief goede amateur-genlocker met tal van krachtige en creatieve mogelijkheden. Verwacht echter geen wonderen van deze genlocker. Het geringe kwaliteitsverlies (high frequency details) van het omzetten van RGB naar PAL-composiet blijft zichtbaar. Hetzelfde geldt bij opname op gewone VHS-videorecorders en vertoning op de huiskamer-TV. Deze achteruitgang ligt niet aan de genlocker, maar is inherent aan de kwaliteit van de overige randapparatuur. Met SVHS-apparatuur en Super High Grade videobanden kunnen de resultaten wat opgepept en de kopieerverliezen beperkt worden
De A8802 kan het signaal van een VCR of Camrecorder goed tracken zolang dit van een redelijke kwaliteit is. Een matige bandkwaliteit en versleten of vuile recorderkoppen geven een slecht resultaat. Hetzelfde geldt voor videotrucages waarbij niet-standaard-synchronisatiesignalen gebruikt worden.
Afgezien van de hier gemaakte kanttekeningen biedt de A8802 redelijk veel waar voor zijn geld. Zeker in deze prijsklasse.

U.S

Voor informatie: Cat & Korsh International, Evertsenstraat 5, 2901 AK Capelle a/d IJssel.

De zon schijnt, de computer staat op het tuintafeltje. Een angstaanjagend gevoel bekruipt je. Je wilt wel een routine schrijven maar je beheerst de taal niet goed genoeg!!! Angst, paniek en ontreddering zijn daar.... maar wat ziet uw oog? Nee het is geen vliegtuig, geen raket of een ballon, nee het is de SUPERAMIGA C CURSUS!!!!

AMIGA C



Wat zullen we deze keer eens doen, 50 pop of een envelop. Nou geeft u mij maar een envelop. Dat kan. Wat voor nummer had u willen hebben? Nummer 61, dat is namelijk de leeftijd van mijn man en mij opgeteld. Oh, wat leuk, dan heeft u zeker nog geen kinderen. Nee, die hebben..... HO STOP, we dwalen heel erg ver af. Na deze aflevering gaan we het dan eindelijk hebben over het onderwerp waar de titel van deze cursus voor staat, AMIGA C. Nu zijn we nog gedwongen enkele noodzakelijke C begrippen te behandelen anders ziet u straks de bomen niet meer door het verzuurde bos.

SWITCH en CASE

De vorige keer waren we nog bij de besturingsstromen. Nu nog even een staartje hiervan. SWITCH en CASE zijn functies waarvan in BASIC geen equivalenten zijn te vinden. De ON/GOTO (GOSUB) mogelijkheid komt nog het dichtste bij. Met SWITCH/CASE biedt de mogelijkheid zich aan op een simpele wijze een programmaloop te beïnvloeden. Aan de hand van een aan de SWITCH functie doorgegeven variabele wordt met een CASE functie een conditie getest. Pfff, wat een rotzin. Het komt er dus simpelweg op neer dat bij SWITCH aangegeven wordt op wat voor variabele en waarde getest moet worden, bij de CASE functie(s) zet u dan een aantal mogelijke waarden op een rij. Een voorbeeld lijkt ons nu illustratiever dan deze hele woordenbrij.

```
main()
{
 int a;

 scanf("%2d",a);

 switch(a)
  {
   case 1:
        print("U heeft de
waarde 1
ingevoerd\n");
      break;
    case 2:
      printf("Goh, weet je,
ik heb een 2
aangetroffen!!!\n");
      break;
    default:
       printf("Nou, ik weet
1 ding, u heeft
geen 1 of een 2 inge-
voerd!\n);
       break;
   }
}
```

Wederom een voorbeeld dat blaakt van de simpelheid. Het zal wel aan het weer liggen of zo. Ook hiervan hebben we een BASIC versie proberen te ritselen. En zoals gewoonlijk lukte dit (met moeite) weer.

```
INPUT a

ON a GOTO label1,label2
PRINT "Nou ik weet 1 ding,
u heeft geen 1 of een 2 in-
gevoerd!"
END
label1:
PRINT "U heeft de waarde 1
ingevoerd"
END
label2:
PRINT "Goh, weet je, ik heb
een 2 aangetroffen!!!"
END
```

Zoals u ziet kost het in BASIC nogal wat moeite om dezelfde programmastroom te realiseren. De programmastroom moet afgebroken worden nadat de PRINT uitgevoerd is. In BASIC staan hier een tweetal comman-

do's voor. het eerste is definitief. Het END commando. het zorgt ervoor dat de programmastroom, BATSSS!, afgebroken wordt. Als tweede is er dan nog de GOSUB/RETURN combinatie waardoor het nog mogelijk is om in het programma te blijven. In C is dit iets eenvoudiger. Het BREAK commando zorgt ervoor dat het binnen de accolades '{' en '}' aanwezige deel routine wordt onderbroken. De programmaloop gaat naar het eerstvolgende hogere niveau, dus in dit geval zijn dat de accolades die het 'main()' programmablok aanduiden. Als laatste treffen we dan nog de 'default case label'. Als deze binnen het SWITCH blok staat, dan zal het programmablok dat bij het 'default case label' hoort, uitgevoerd worden als geen van de overige CASES uitgevoerd kon worden. Het is dus een DEFAULT!

GOTO

Ook in C bestaat het commando GOTO. Toch neemt deze gelijknamige tegenvoeter van het veelgebruikte BASIC commando niet zo'n prominente plaats in. De taal C biedt over het algemeen betere middelen om een programma te schrijven. Toch kan het soms voorkomen dat het commando persé nodig blijkt te zijn. Bijvoorbeeld om uit een binnenste lus van WHILE/IF/SWITCH CAST constructie naar een van de buitenste lagen te gaan. Goed, in dit geval is het commando gerechtvaardigd. Alle overige smoezen keuren we zondermeer af. Het commando GOTO heeft een algemene syntax:

```
GOTO [labelnaam]
```

Bij de labelnaam dient u er op te letten dat er een dubbele punt (:) achter gezet moet worden. Een voorbeeld hiervan zullen we niet geven, wel een algemene illustratie.

```
main()
{
 if(.....)
   {
     . . . . . . .
     goto sprong;
     . . . . . . .
   }
sprong:;
}
```

Functies

We spreken al afleveringen lang van functies in plaats van commando's. In feite is dit niet geheel juist. Je spreekt van een functie als van het programmadeel een waarde terug verwacht mag worden. Anders heet dat blok een procedure. In geen geval kunnen we spreken van een commando. Dit laten we dus ook volledig achterwege. We zullen het alleen gebruiken als het echt van toepassing is. Goed, de functies en procedures. Het is in C mogelijk zelf functies en procedures aan te maken. Dit is zelfs aan te raden daar het de leesbaarheid van een programma sterk vergroot. Hoe krijgen we dit nu voor elkaar? In C worden procedures en functies op nagenoeg de zelfde wijze in elkaar gefrutseld. Dit in tegenstelling tot de taal PASCAL. Hoe maken we nu een procedure of een functie? Vrij simpel. U bedenkt een naam. U geeft de variabelen aan die de functie of procedure nodig heeft en u geeft aan wat voor type deze variabelen zijn. Wat is dan het verschil tussen het opstellen van een functie en een procedure. Dat komt nu. Daar een functie een waarde terug moet leveren, zult u aan moeten geven van wat voor type deze waarde moet zijn. En voordat u de functiebody (het geheel met de accolades, weet u nog wel?) afsluit, moet u de waarde aan de hoofdroutine terug geven. Dit doet u met behulp van de RETURN() functie. In fig. 1 ziet u de algemene structuur van zo'n procedure.

U ziet, als u eenmaal een functie of procedure aangemaakt heeft, kunt u hem altijd weer aanroepen in uw programma met de naam die u de procedure of functie gegeven heeft. Nog even een opmerking omtrent het 'casten' van de functies. Standaard zal de compiler aannemen dat de teruggegeven waarde van het type INT zal moeten zijn. Dus als u niets vermeld zal de teruggegeven waarde van het type INT zijn. In dit voorbeeld hebben we het alleen maar er voor vermeldt om aan te geven dat hiervoor de 'type casting' plaats zal moeten vinden. Nog even voor de volledigheid, 'type casting' zal u moeten doen als een variabele van het ene naar het andere type moet worden omgezet. Stel u heeft een variabele 'a' die van het type INT is en u wil een berekening met 'floating point' getallen uitvoeren dan zou u de variabele als volgt kunnen 'casten',**(float)a**.

Goed, een opmerking voor de Aztek compiler gebruikers. Na enige ervaring is ons gebleken dat functies die vanuit andere procedures of functies aangeroepen worden, boven deze functie of procedure vermeld moeten staan. Doet u dit niet, dan zal er een compiler-foutmelding volgen. Zo, weer even terug naar de theorie. Variabelen die u binnen een functie of procedure declareert zullen 'lokale variabelen' zijn. Dit wil zeggen: deze variabelen verliezen hun waarde als de functie of de procedure verlaten wordt. Een compiler zal de in een andere procedure of functie gedeclareerde lokale variabelen niet herkennen en vermelden dat de variabelen niet bekend zijn. Een oplossing voor dit probleem biedt zich aan in de vorm van 'globale variabelen'. Dit zijn variabelen die NA de INCLUDE'S en VOOR de eerste functies of procedures vermeld worden. Deze variabelen behouden hun waarde door het gehele programma. Toch is het verstandig niet al teveel variabelen globaal te laten zijn, daar dit veel geheugenruimte in beslag neemt en het niet van nette programmering getuigt. Een voorbeeld van lokale en globale variabelen.

```
Procedure:
[naam](variabele1,variabele2,etc..)
[type] variabele1;
[type] variabele2;
{
  (De eigenlijke procedure)
}

Functie:
[type][naam](var1.,var2.,etc..)
[type] variabele1;
[type] variabele2;
{
 (De eigenlijke functie)
 return(WAARDE);
}
 De doorgekregen waarde is 10 Ontvangen waarde :27
```

fig. 1 algemene procedure

```
#include "stdio.h"
 int a;   /* globale varia-
bele *
 main()
 {
  int b,c; /* lokale varia-
belen */

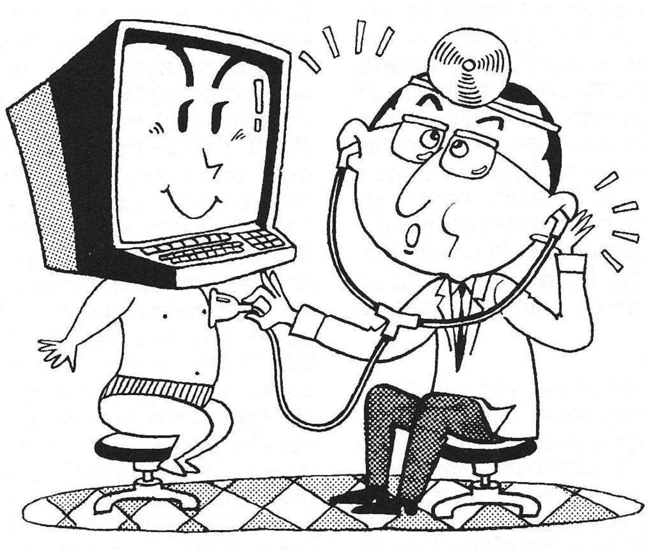
  a=17;
  b=c=25;

                  printf("a,b,c
:%2d,%2d,%2d\n",a,b,c);
  test();
 }

 test() /*let op de fout in
printf() */
 {
                  printf("a,b,c
:%2d,%2d,%2d\n",a,b,c);
 }
```

De INCLUDE'S

We hebben het er daar net al over gehad, include statements. Dit is weer een van de aantrekkelijke eigenschappen van de taal C. Met een INCLUDE statement is het mogelijk bepaalde programmagedeelten in uw eigen programma te betrekken. Stel u gebruikt vaak een routine dan zou u deze apart op schijf kunnen zetten. Voordat uw eigenlijke programma begint, haalt u deze routine dan weer in het programma met behulp van het INCLUDE statement. In Amiga C wordt veelvuldig van include's gebruik gemaakt. Dit alleen al vanwege het feit dat de Amiga veel functies en structures nodig heeft om al haar eigenschappen te kunnen beinvloeden. Een include statement begint altijd met het teken '#' en vervolgens 'include' met daarachter tussen haakjes of



groter-dan- en kleiner-dan-tekens de naam van de file.

Een C compiler zal dit voorbeeld niet slikken. Immers, in de procedure 'test()' worden in de printf() functie de variabelen b en c aangeroepen. Deze variabelen zijn niet in de procedure 'test()' gedeclareerd maar in main(). Dus het zijn lokale variabelen, immers main() is ook een functie. Alle binnen de accolades gedeclareerde variabelen zijn lokaal. Een oplossing voor het bovenstaande voorbeeld is b en c ook globaal te maken of de waarden van de variabelen in de parameter-lijst van de functie test() door te geven, bijvoorbeeld; **test(b,c);**. De procedure 'test()' moet dan ook een aanpassing krijgen.

```
 test(r,s)
 int r,s;
 {
 printf("a,(b=r),(c=s)
:%2d,%2d,%2d\n",
 a,r,s);
 }
```

Structure's

In het voorbeeld van de vorige aflevering, dat deze keer is afgedrukt, werd veelvuldig gebruik gemaakt van structures. Wat zijn nu structures? Structures zijn wat men in de taal PASCAL onder de naam RECORDS verstaat. Een structure is dus een verzameling van losse gegevens die onder 1 noemer bereikbaar zijn. Stel, u schrijft een adressenbestand. Om er nu voor te zorgen dat u met uw gegevens 'netjes' in het programma om kunt springen, kunt u besluiten om een aparte structure te maken. Deze structure zou er als volgt uit kunnen zien.

```
 struct adres {
  char naam;
  char adres;
  char postcode;
  char woonplaats;
 };
```

Het is nu mogelijk onder de noemer 'adres' de aparte gegevens 'naam', 'adres', 'postcode' en 'woonplaats' te bereiken. Nu zult u zich afvragen, hoe dan? Juist, hiervoor zijn twee aparte operatoren. De zogenaamde puntoperator, '.' en de pijloperator '->'. Welke operator u gebruikt is afhankelijk van de wijze waarop de variabele gedeclareerd wordt. In dit geval zou u de naam als volgt uit kunnen lezen,**adres.naam**. Als u nu meer gegevens in zou moeten voeren, wat op zich wel handig is bij een adressenbestand, dan declareert u in plaats van de ene naam adres een array van het structure adres. Het volgende voorbeeld illustreert dit.

```
 struct adres {
   char *naam;
   char *adres;
   char *postcode;
   char *woonplaats;
 } vermelding[100];
```

Nu kunt u onder de noemer 'vermelding' weer alle adres gegevens bereiken. Aangezien het hier om een array gaat, kunt u dus 100 keer, oftewel van 100 personen, gegevens opslaan. Het kan ook voor komen dat u een eigen structure in een INCLUDE file opgeslagen heeft. Hierbij is het persé noodzakelijk dat u de noemer 'adres' vermeldt. Bovenstaande structure kan namelijk ook in een programma als volgt gedefinieerd worden.

```
 struct {
   char *naam;
   char *adres;
   char *postcode;
   char *woonplaats;
 } vermelding[100];
```

Dit zal in een programma hetzelfde effect hebben als bovenstaand voorbeeld. Er is echter een groot verschil. Als u het net gegeven voorbeeld in een INCLUDE file in zou voeren, zou het netto resultaat 'njente' zijn daar u de structure niet meer kunt bereiken voor latere declaraties. Dus de naam 'adres' achter 'struct' dient ervoor om latere declaraties mogelijk te maken. Zo'n declaratie zou er als volgt uit kunnen zien.

```
 struct    adres    vermel-
ding[100];
```

Nog even ten overvloede, een dergelijke declaratie kan alleen maar voorkomen, als de structure al eerder bekend gemaakt is aan het systeem.

Het hernoemen van een structure gaat dus als volgt. struct [oude naam] [nieuwe naam];
Een meer op de Amiga toegespitst voorbeeld zal mischien nog verhelderender werken. Het gaat hier om de Window structuur die gedefinieerd wordt in de INCLUDE file intuition/intuition.h".

```
struct Window *MijnWindow;
```

De voor de naam 'MijnWindow' vermelde ster dient er voor om te zorgen dat zonder veel moeite later het adres naar het nieuwe window in de variabele MijnWindow opgeslagen kan worden. In dit geval zult u gebruik moeten maken van de pijl operator '->'. Nu is dus de gehele Window structuur onder de naam MijnWindow te bereiken.

Ter afsluiting

Tot zover deze aflevering. Weliswaar niet zo lang als u gewoonlijk gewend bent, maar we zijn nu door de algemene beschrijving van de taal C heen. Er is natuurlijk nog veel meer te zeggen over de taal C, maar daarvoor verwijzen we u toch door naar gespecialiseerde boeken over de taal C. Een boek dat ons vaak terzijde stond, staat en zal staan is hieronder vermeld. In de volgende aflevering gaan we in op wat het eigenlijke doel van deze cursus is, C toegepast op de Amiga computer. Als eerste zullen we ingaan op het grafische besturings systeem van de Amiga, Intuition. Tot de volgende keer!



Literatuur:

De programmeertaal C, Grondbeginselen en toepassingen, Al Kelley & Ira Pohl,
Addison Westley,
ISBN 90 6789 068 5

JOHAN & JOHAN

Zuma's TV*TEXT voor de Commodore Amiga is een tekstpresentatie-programma voor videoproducties, computer-slideshows en tekstdia's. De software maakt van de meegeleverde en optionele Amiga-fonts (lettertypen) werkelijk magnifieke titels. Variaties in lettergrootte, schaduwwerking, semi-3D-effecten, letterkleur en shaded achtergronden brengen een professionele aftiteling in huis. Geen tekst-speelgoedje voor op de computer maar een volwaardige titelgenerator die qua mogelijkheden professionele titelhardware evenaart of zelfs klopt.

TV*TEXT

Professionele video-titels op de Amiga

Titelgeneratoren zijn een must voor hen die professioneel uitziende videoproducties, kabelkranten, slideshows en tekstdia's willen maken. Voor video zijn tal van losse titelgeneratoren leverbaar en een aantal camrecorders beschikt al standaard over een ingebouwde aftiteloptie of beeldgeheugen waarmee titels en afbeeldingen aan de gemaakte opnamen kunnen worden toegevoegd. Een tweede veelzijdiger mogelijkheid is het gebruik van een PAL-compatible Personal Computer zoals de Commodore Amiga 500, 1000 of 2000. Die Amiga is tevens geschikt voor het schieten van tekstdia's voor conventionele diapresentaties en slideshows.

Bij puur videogebruik zult u over een PAL-composiet-videokaart of genlock dienen te beschikken. De PAL-kleurenkaart zorgt voor vertaling van de RGB-signalen uit de Amiga in een composiet-videosignaal voor de videorecorder. Een genlock maakt overlaying, d.w.z. het over elkaar heenleggen van twee videobronsignalen (bijvoorbeeld de gemaakte titels en de opname van de camrecorder) mogelijk.



Softwarematige titelgenerator

Een video-computer zoals de Amiga beschikt al standaard over de benodigde grafische chips om een gedetailleerd scherp kleurenbeeld op de monitor te zetten. Animatiechips ondersteunen de vloeiende bewegingen van de grafische beelden. Bij een programma als TV*TEXT 1.11 gaat het echter voornamelijk om fontbewerking.

Een font is een variatie van een bestaand lettertype (typeface), bijvoorbeeld Swiss, Courier, Times, Roman, Pica, Barn en Diamond. Die variaties betreffen meestal de grootte uitgedrukt in points (honderste delen van één inch), vetten (bold), onderstrepen (underline) en schuinschrift (Italic). De drie laatst genoemde fontvarianten worden ook wel stijlen (styles) genoemd.

Fonts worden standaard bij een aantal pogramma's (Amiga Werkbank, tekstverwerkers, Business Graphics) meegeleverd. Ook zijn zij als losse libraries (letterbibliotheken) op diskette leverbaar. Zuma zelf levert ook een aantal font-bibliotheken op Amiga-schijfjes.

Zonder verdere bewerking zijn fonts een wat statisch geheel en spreken weinig in dia- en videoproducties. Anders wordt het als we de kleuren, schaduwwerking, belichting, mate van rotatie en achtergrond kunnen beïnvloeden. Dan ontstaan professionele titels die op menig kabelstation niet zouden misstaan. Dat nu precies wat een titelgenerator zoals TV*TEXT doet. Neem een bestaand font en maakt daar met behulp van de keuzemenu's en muis een flitsende titel mee.

TV*TEXT gebruikt drie Amiga screenmodes:

- De 640 x 200 pixels medium resolution, het gemiddelde oplossende vermogen goed voor normaal video- en diagebruik
- De 640 x 400 pixels high resolution, HIRES videomodus voor het scherpst mogelijke resultaat.
- De 704 x 204 pixels overscan geeft full size video op het monitorscherm. Dus geen last meer van storende zwarte randjes in het videbeeld bij opname op een VCR. Bedenk wel dat deze videomodus video-RAM-geheugen vreet. De mate van overscan is regelbaar via de SCREEN SIZE-optie van het videomenu

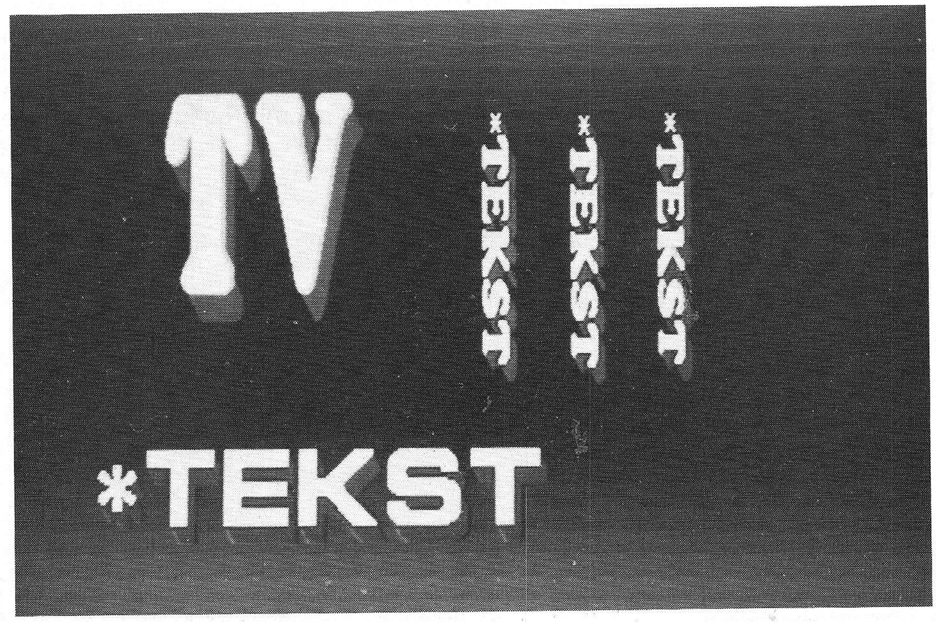
Die 640 x 400 pixels HIRES mode is interlaced. Dat wil zeggen, het videobeeld wordt in twee keer opgetekend. Eerst de oneven horizontale lijnen en dan de even beeldlijnen. Interlacing is noodzakelijk om compatible met de bestaande videosystemen te blijven. Het nadeel is een lichte flikkering die dankzij het nagloeien van de fosfordeeltjes op de TV-beeldbuis niet voor het menselijk oog zichtbaar is. Bij het fotograferen van de beeldbuis kan het interlace-effect bij een te snelle sluitertijd wel zichtbaar worden! Experimenteer daarom eerst bij verschillende sluitertijden (bijvoorbeeld 1/8, 1/15 en 1/30 seconde) om de best dekkende tijd te vinden.

Gerender

TV*TEXT maakt gebruik van het zogenaamde RENDER-principe. Het Angelsaksische woord "render" staat voor interpreteren of presenteren. Bij de meeste tekenprogramma's tekent u de tekens en figuren ter plekke op. De bewegingen van de muis of tekenpen bepalen on the spot wat er op het scherm gebeurd. Dit in tegenstelling tot Zuma's TV*TEXT waarbij de gebruiker een eerst ingetypte tekst door de keuze van de programmamenu's laat bewerken.
In de praktijk komt dat neer op het kiezen van de NEW TEXT-optie uit het RENDER Menu waarop het tekstinvoervenster in beeld verschijnt. Zoek vervolgens een geschikt font met bijpassende lettergrootte uit bij het textmenu en kies de letterkleur plus de gewenste special effects. De tekst blijft gewoon zwart, maar wel in de juiste fontvorm en grootte op de invoerregel staan, totdat u de "rendertoets" met de linker muisknop activeert en vervolgens de tekst het vrije scherm opsleept. Daar tekent TV*TEXT de titel volgens de opgegeven specificaties. Transport naar de gewenste locatie gaat via de muisbewegingen en met een druk op de rechter muisknop wordt de tekst ter plekke gedumpt. Bij meerdere regels of een te grootte tekst zult u de titel in delen moeten aanbrengen en dat vergt nogal wat gerender voordat het allemaal op zijn plaats staat. Om over eventuele fouten nog maar niet te spreken.
Behalve tekst biedt TV*TEXT ook een beperkt aantal shapes. Een shape is een geometrisch figuur zoals een cirkel, lijn of rechthoek. Ook de shapes kunnen via het rendermenu van de nodige special effects voorzien worden. Het tekenen gaat direct op het scherm met behulp van een kruishaar-cursor.

- De effectdiepte bij de schaduw- en stroboscoopeffecten
- De letterstijleffecten Edged, Outlined en Extruded (3D-profiel)
- De kleur van de objectfront
- De profieldiepte
- Rotatie in stappen van 45 graden: Staat onder het text-menu

Een aparte techniek vormt de overlay rendering. Daarmee kan de gebruiker met andere tekenpakketten gemaakte objecten binnen TV*TEXT halen en met de render special effects bewerken. Bij met name videotoepassingen geven een zwarte outline, schaduweffecten en een 3D-profiel respectievelijk meer scherpte en diepte.



De special effects

Met behulp van de speciale effecten brengt TV*TEXT leven in uw video- of diatitels. Deze special effects zijn toegankelijk via de RENDER PREFERENCES REQUESTER. Er is keuze uit:

- Het besturen van de schaduw-, strobe- en stijllichtval
- Het veranderen van de schaduw- en stroboscoopkleuren
- Keuze uit de schaduwtypen Drop (schaduw staat los van het object), Cast (schaduw vast aan het object), Strobe (= hier een veelkleurige schaduw) en Transparent (doorzichtige schaduw)

Achtergrond

Het spreken van een titel staat of valt met de juiste achtergrondkeuze. Behalve een goed kleurcontrast kan een juiste kleurgradiënt de titel meer impact geven. De achtergrondkleur wordt aan de hand van het beschikbare kleurenpallet ingesteld. Er zijn drie mogelijkheden: keuze van een vaste kleur uit het pallet, instellen van de Rood-, Groen- en Blauw-regelaars en de Hue (tint)-, Saturation (kleurverzadiging)- en Value (helderheids)-regelaars. Met behulp van Range valt een fade-overgang tussen twee verschillende kleuren te maken. Swap en Copy kunnen de kleuren binnen het pallet verplaatsen. De optie Alt Pallette wisselt een gegeven alternatief pallet uit met het huidige pallet. En tot

slot verzorgt Auto Color de automatische kleurstelling bij het laden en renderen van plaatjes uit andere grafische pakketten.
De gradation verzorgt kleurfading in 16 stappen. Verder biedt de achtergrond nog de keuze uit verschillende rasters (de Grid-optie) en behangselpapier (Tile & Wallpaper, met eerder opgeslagen motieven).

Editing

Onder de redigeeropties staan de commando's voor de titelopmaak. UNDO maakt de laatst verrichte handeling ongedaan. CUT verplaatst een uitgesneden beelddeel naar een andere schermlocatie. COPY doet het zelfde met de uitsnede maar maakt tevens kopieën van het origineel.
Tot de vormveranderende opdrachten behoren o.a. RESIZE (groter/kleiner), SQUEEZE & STRECTCH (samenknijpen en uitrekken), HALF (halve grootte), DOUBLE (dubbele grootte), DOUBLE HORIZONTAL, DOUBLE VERTICAL en HALF VERTICAL.

TV*TEXT is een goede titelgenerator voor videofilms, slideshows en titeldia's. Met een Amiga reeds op het bureau bent u dan veelzijdiger en vaak goedkoper af dan met een losse titelgenerator voor video of business graphicspakket. Het gerender is in het begin wat ongemakkelijk en met name de rotaties duren tamelijk lang. De prijs van *f* 279,- vinden wij iets aan de pittige kant, vooral omdat daar in de praktijk nog wat extra fonts bij zullen komen.

Informatie bij o.a.: CAT & KORSH International, Evertsenstraat 5, 2901 AK Capelle aan de IJssel.

U.S

Kleine advertenties

Te koop: C16

Met datarecorder, Ned. handl., Basic intro cursus. *f* 100,— S. de Jong, Tel. 05105-2893.

Te koop C 128D + diskdrive

120 disks + veel boeken + uitbreiding 1750 Ram + Final cartridge + software + 2 joysticks, 2 floppybakken. *f* 1.200,—. Tel.: 02263-1941

Te koop Commodore printer

Color MCS 801 voor C64/128 met kabel. Vraagprijs: *f* 275,—
Tel.: 01824-2466

Te koop: Amiga 500 + 501

2e drive, T.V.adapter en joystick, Philips 8833 kl.st.mon., Disks - workbench + Amiga Basic + extra's. Veel spellen met tot. waarde *f* 3.800,—
Eén koop *f* 2.750,—.
Tel.: 02975-60515.

Gezocht:

Commodore disk drive C1571 voor aansluiting als tweede drive op de C128 D. Tel. na. 19.00 uur: 023-276317.

Te koop: Commodore 64

Met 1541 Diskdrive, C2N recorder, joystick, snellaadcartridge,
Expert ESM, vele diskettes en cassettes en boeken. Vaste prijs *f* 850,—
Tel. 075-312619 (na 16.00 uur).

Gevraagd: Intromakers

of intro-linkers. Op disc, voor CBM-64. Tegen betaling of ruilmateriaal.
Tel.: (na 15.00 uur) 079-165670 (Rik).

Te koop: CBM-64 set

+ 1541 diskdrive + MPS-801 printer + Philips monitor + cassetterecorder + joystick + programma's (o.a. GEOS, Flight Sim II) + diskettes + bijbehorende lit. voor *f* 995,—
Tel.: 070-979721 na 19.00 uur.

Te koop: printer.

Apple II C printer.
Geschikt voor Commodore 64, prijs *f* 200,—. Tel.: 02940-12124.

Gevraagd power-cartridge!

voor CBM-64. prijs: ca. *f* 25,—
Tel.: 050-412411

Te koop: compleet systeem

CBM-64 + 1531 recorder-1541 diskdrive-Sharp kl. t.v.-, Star NX-10C printer - modem, 2 joysticks - software, Powercartridge + lichtpen
Prijs *f* 2.000,—
Tel.: 01804-12814 (na. 16.00 uur).

Gezocht:

Iemand die Little Computer People(s) voor de C-64 heeft, en met mij wil ruilen. Evt. ook andere software
Tel.: 055-336783

Te koop:CBM 1280 PC

(d.w.z. C64 + C128 + ingebouwde diskdrive) met 300 schijven, zijn 2000 spelen in 2 diskettebakken en 2 joysticks + 15 boeken en handleidinkjes, o.a. machinetaal en evt. een plekkie in onze groep toyboys. Verkoop wegens aanschaffing PC. *f* 1.200,—
i.z.g.s.
Tel.: 01748-15879

Gevraagd drive voor C-64

Ook te koop: Originele spelen op tape voor C-64 (Gunship, op wolf...enz...)
Tel. 055-412393

Gevraagd: pen & cassette

Wie kan mij helpen aan pen en cassette voor plotter 1520. Mag leeg zijn, moet dienen om na te maken.
P.J. Michilsen. Tel.: 055-213822

Gezocht:Nederl. handl.

voor GEOS tegen vergoeding
Tel. 085-257339.

Te koop voor C-64

(originele versies op cass met handl.): laser genius (assembler/disassembler/debugger/analyser); laser basic (spelontwerp); assembler cursus: spritemachine + champ assembler; 64-intern(boek); gunship (helicopter simul); music composer; arendarvon castle (adventure+boek).
Tel. zaterdag na 18.00 uur: 011/611194(België).

'The Untouchables' op diskette. Chicago in de jaren '20, maffia, illegale drankimport en straatbendes. In de broeierige hitte, 'downtown' waar de regels van de straat gelden. Dàt is Capone, een action-game waarbij gleufhoeden en grote limousines noodzakelijk, maar niet voldoende zijn om op de been te blijven.

Capone

Jaren '20 herleven in actiongame

Capone is een spel van de software ontwikkelaars Actionware. De naam geeft eigenlijk al het type spel weer, het is allemaal ACTIE wat de klok slaat. Het spel is een nieuw soort actiespel-adventure, het speelt zich af in een atmosfeer van een adventure, maar elke onoplettendheid wordt onmiddellijk afgestraft.

Het spel speelt zich af in het verre Chicago in de jaren 20. Het is een roerige tijd, gangsters regeren op de straten, zij waren het die de regels hier opstelden. Winkeliers, bar-eigenaren, alles moest zich onderwerpen aan dit gezag. Wil men hier kunnen overleven, dan moet er betaald worden, zo niet dan .....

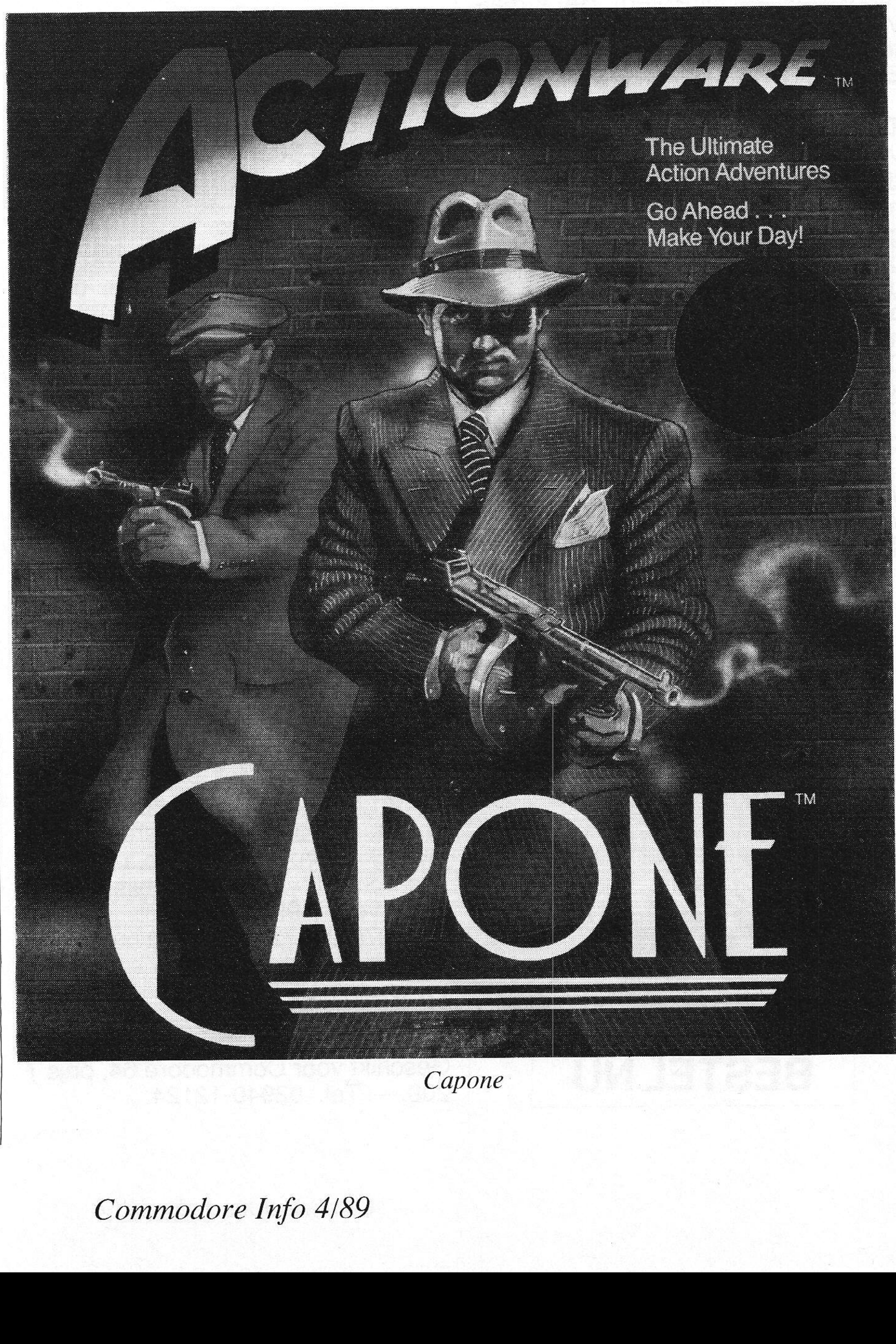
Als ook dat niet helpt, moeten er andere maatregelen worden getroffen, waarbij ook onschuldige burgers niet veilig zijn. U moet als geheim agent deze misdaden proberen te bestrijden, en er rest niets anders dan: veel succes!

Pistool

Amiga-gebruikers zijn gewend aan het werken met de muis, een aantal spelen wordt met een joystick gespeeld, ook geen probleem. Dit spel heeft iets unieks: het spel kan ook worden gespeeld met een pistool. Als extra kan deze met het spel worden meegeleverd. Het is een geweldige ervaring, met een pistool, aangesloten op de muispoort, op het Amiga-scherm te kunnen schieten en dan de gangsters te zien vallen. Gebruik alleen het originele Capone pistool, we kunnen niet instaan voor de gevolgen als U een echt pistool gebruikt.

Onmogelijke opdracht

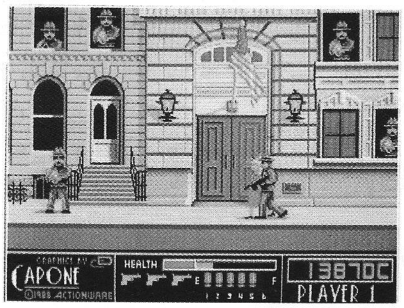
Na het intro-scherm moet er een keuze worden gemaakt, of er met een muis of met een pistool wordt gewerkt. Het verdient aanbeveling het spel eerst een paar keer met de muis te spelen om wat ervaring op te doen.

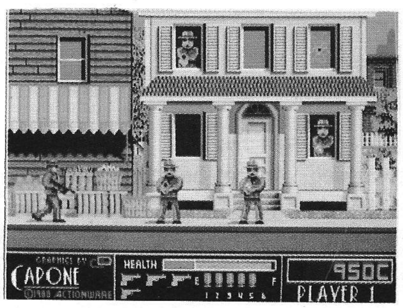


Capone



Dit omdat zeker in het begin het pistool het wat moeilijker maakt. De tweede keuze die gemaakt moet worden is: speel je het alleen of met twee personen. Het spel kan gespeeld worden in verschillende moeilijkheidsgraden. De eenvoudigste is cadet, hierbij zijn geen kinderen, honden en dynamiet. Bij Rookie wordt dit al aanmerkelijk slechter. Captain is een keuze die een nagenoeg onmogelijke opdracht inhoud. Kies eerst maar eens voor cadet. Wordt het spel met twee personen gespeeld dan kan een ieder op een ander niveau spelen. Nu begint het spel pas echt. Je kijkt vanuit je schuilplaats naar een vredige, rustige straat. Al gauw blijkt dit maar schijn te zijn. Ramen gaan aan diggelen, achter elke kist, vanuit elke hoek vanuit de ramen overal wordt er op je geschoten. Dit is alleen te voor komen door sneller te zijn dan je tegenstander, dus sneller schieten. Let op de gangsters dragen allemaal een hoed, de personen die deze niet dragen zijn "onschuldige" burgers, het kost je veel punten als je deze ook raakt. Daarom niet te snel schieten, maar eerst goed kijken. Heb je in het begin de keuze cadet, dit is het eenvoudigste, dan komen de gangsters ook de straat van links en rechts inlopen. Heb je hier alle gangsters verslagen, dan kan je verder doordringen in de door hun beheerste wijk. Nu sta je in een straat bij een bakker en een kapperszaak.





Dynamiet

Nu wordt er, behalve dat er van alle kanten wordt geschoten, ook met staven dynamiet gewerkt. Zodra er staven neergelegd worden moet hier op worden geschoten, dit maakt de staaf onschadelijk. Let ook goed op de dozen en kisten aan de zijkant van de gebouwen. Hoe verder je doordringt in deze wijken, des te moeilijker wordt je opdracht. De gangsters komen in steeds grotere aantallen, en ze volgen elkaar steeds sneller op. Probeer zuinig te zijn met de munitie, je bent er sneller door dan je denkt. Kom je ook hierdoor heen dan speelt het volgende gevecht zich in een loods af. De tegenstanders verschuilen zich hier achter kisten. Burgers zijn hier niet, dus er kan op alles geschoten worden wat beweegt. Neem dit maar letterlijk, want de kisten zitten vol dynamiet, deze mogen absoluut niet geraakt worden.

Pas op

Nadat je de orde in deze hal hebt terug gebracht, moeten we buiten nog kijken of er werkelijk niemand is ontsnapt. Hierbij is het wel belangrijk op burgers te letten. Tijdens de vuurgevechten verschijnen er af en toe pistolen in het beeld. Door hierop te schieten krijg je er een leven bij. Nu blijkt alles wat hiervoor gebeurt is maar kinderspel te zijn, alle zeilen moeten worden bijgezet om het volgende niveau te overleven. Over kinderen gesproken, zij zijn het die op rollerskates je opdracht nog moeilijker maken door tijdens een vuurgevecht door het beeld te komen. Raken wordt direct afgestraft. Let op, want niet alles is wat het lijkt, dit lijkt moeilijk, maar je komt hier gauw genoeg achter. Bomen in de straten zijn ook veel gevaarlijker dan je op het eerste gezicht zou denken. Ze blijken wat met dynamiet te maken te hebben.



Punten

Punten worden verkregen door het neerschieten van de gangsters. Om hoog op de scorelijst te komen moeten er dus veel worden neergeschoten. Maar om echt hoog te komen moet het "mystery" worden geraakt. We vertellen je alleen dat deze zich ophoudt in de buurt van het postkantoor. Wordt het je tijdens het spelen allemaal wat te veel of wordt je gestoord, bijvoorbeeld de telefoon, dan kan het spel gestopt worden door op de spatiebalk te drukken.

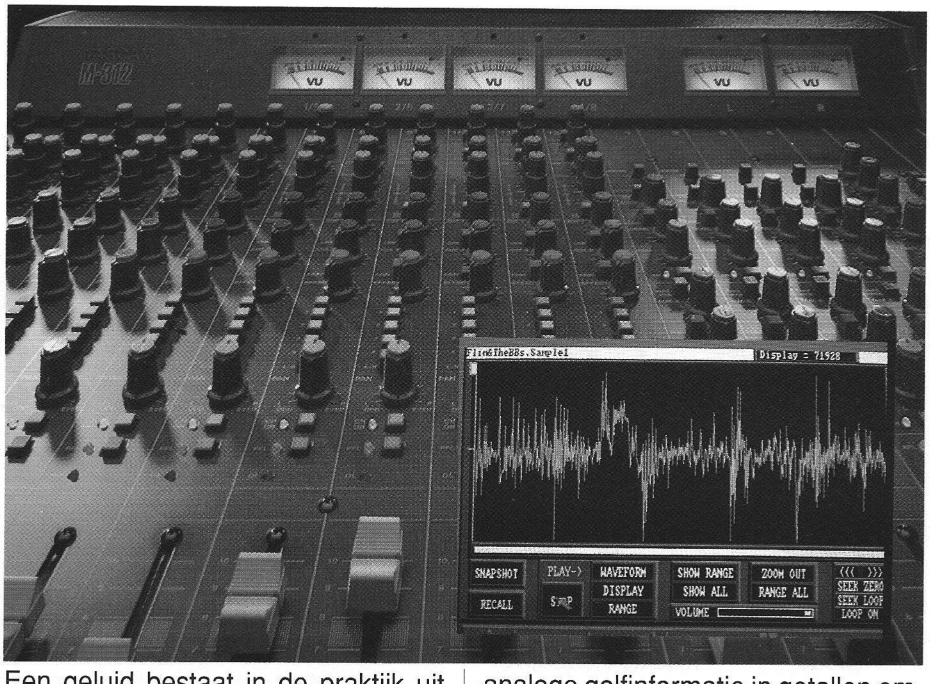
De Amigafamilie beschikt over een krachtig digitaal systeem voor het nemen van geluidsmonsters. Met behulp van digital sound sampling kan de gebruiker real-life geluid omzetten in data voor opslag en/of verdere bewerking door de Amiga. Nodig zijn een geluidsbron (microfoon, recorder, CD- of videoplayer), sampling hardware en software. Audiomaster II van Aegis is de verbeterde versie van dit bekende stereo sound sampling-pakket. Daarmee verandert u de Amiga in een geavanceerde geluidsstudio voor digital sound processing.

Audiomaster II

Geluidsnoepen en digitaal bewerken op de Amiga

Full stereo digital sound sampling is een wat minder bekende toepassing van de Amiga. Onbekend maakt ook onbemind, maar daarnaast dien je als gebruiker ook over de nodige expertise te beschikken om echt iets met gedigitaliseerd geluid te kunnen doen. Aegis biedt met Audiomaster II gebruiksvriendelijke software voor het zelf verder opslaan en bewerken van gedigitaliseerd geluid. Zelfs de niet-expert kan er zo mee aan de slag.

Geluid beweegt zich voort als een golf. Waarschijnlijk kent u het begrip geluidsgolf nog wel van de natuurkundeles op school. Daar had de leraar het over longitudinale en transversale golven, golflengtes, frequenties, amplitudes en sinuscurven. Een geluidsgolf is in principe **longitudinaal**, d.w.z. de luchtmoleculen vibreren heen en terug in de richting waarin de totale geluidsgolf zich voortbeweegt. Zo'n golf valt moeilijk visueel te beoordelen en daarom geeft men meestal een **transversale**, d.w.z. de deeltjes bewegen in een richting loodrecht op de bewegingsrichting van de golf, sinuscurve. Zo ontstaat de bekende geluidssinus om de normaalas. De uitslag (positief of negatief) ten opzichte van normaalas wordt **amplitude(druk of -volume)** genoemd. De afstand tussen twee golftoppen heeft de **golflengte** en het aantal golfcycli per seconde is de **frequentie** in Hertz. Gaar er al een lichtje van herkenning branden?
Natuurlijk is zo'n eenvoudige golfvorm domweg te mooi om waar te zijn. Muziek- en geluidsgolven blijken bij nauwkeurige analyse behoorlijk gecompliceerd in elkaar te zitten. Hetgeen wij mensen als muziek of een bepaald geluid ervaren is meestal een uit meerdere golven samengesteld geheel. Een pure toon komt in de natuur vrijwel niet voor. Bovendien varieert de geluidssterkte in de tijd, op het moment dat de sinus de normaalas doorsnijdt is de amplitude en daarmee de geluidsdruk nul.



Een geluid bestaat in de praktijk uit een **fundamentele** en de daarbij behorende **harmonische boventonen**. Het aantal en de aard van de harmonische boventonen bepaalt de karakteristiek van de gecombineerde golf die een muziekinstrument voortbrengt. Daarom zijn de geluiden zo verschillend. Zelfs uitgaande van dezelfde fundamentele toon, de **pitch**. Bijvoorbeeld een hoge C.

Digitaal samplen

Net als bij andere vormen van digitaliseren wordt bij **sound sampling** analoge golfinformatie in getallen omgezet. De kwaliteit van het digitaal opgenomen geluid hangt sterk af van hoeveel monsters de sampling hardware per tijdseenheid kan nemen. Dus hoe hoger de **sampling rate**, des te beter de natuurgetrouwheid van het gedigitaliseerde geluid.
Een tweede kwaliteitsbepalende factor is wat de sampler met het zo juist gedigitaliseerde geluid verder kan doen. Bijvoorbeeld storingen en bijgeluiden eruit filteren of fouten en missers bij de opname zelf te korrigeren. Verdere bewerking is een taak van de

sampling software zoals het hier besproken **Audiomaster II**- pakket. Eenmaal gedigitaliseerd geluid kan later weer door de Amiga sound-chip in HiFi stereo worden afgespeeld. Dat heet met een technische term "playbacken". Opmerkelijk is dat Audiomaster II een golfvorm sneller kan afspelen dan de 29 KB sample speed. Dat is meer dan de Amiga-hardware volgens de specificaties zou kunnen halen. Het nut bij andere tragere muzieksoftware is in deze natuurlijk beperkt.

Waarom samplen?

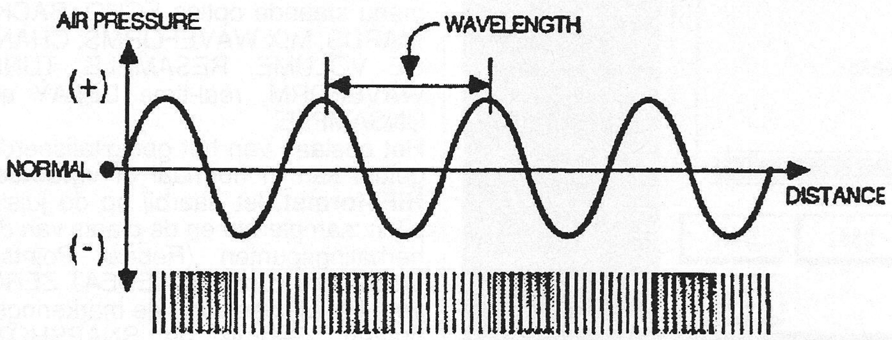
Menige lezer zal inmiddels de vraag stellen: "Waarom zou iemand de Amiga als een soort geluidsinblikker gaan gebruiken?" Daarvoor kunnen diverse redenen zijn. Ik geef een kort overzicht van de meest gebruikte toepassingen:

- Het **integreren van real-life geluid of muziek in videogames**. Tal van de levensechte geluiden van de Amiga- spelletjes zijn op deze manier gemaakt.

Vroeger waren voor dit soort trucs zware computers en dure sampler hardware nodig. De Amiga heeft de electronica voor sound processing al in huis. Resten nog een optionele sampler en de benodigde editing software.

Audiomaster II

Zoals de naam al aangeeft is **Audiomaster II** de opvolger van "geluidsmeester I". Er is een aantal belangrijke verbeteringen aangebracht die volgens Aegis de laatste Amiga-beperkingen bij sound sampling (zullen) slechten.
Als eerste de **verdubbelde sampling rate**. Nu 56.000 samples (monsters) per seconde in plaats van 28.000 sps. Dat is beter dan de doorsnee CD-speler voor elkaar bokst.
De Audiomaster II ondersteunt **full stereo**. Dus is volledig stereo en kan bovendien per geluidskanaal alle gangbare stereo-effecten aansturen. Goed voor meesleepende geluidseffecten zoals afgrijselijke gillen, langsscheurende jets of racewagens, jan-



Eenvoudige sinusvoorstelling van een geluidsgolf

- Het **aanpassen** en/of **korrigeren** van bestaand geluid volgens de eigen specificaties. Bijvoorbeeld het herstel van een slecht origineel.
- Het **zelf creatief componeren** met opgeslagen geluid. Bijvoorbeeld het combineren, vervormen, versterken/verzachten, verandering van de harmonischen en het aanbrengen van special effects. Veel electronische muziek en geluidseffecten voor spelletjes worden op deze manier gemaakt. Gesampled geluid kan door compatible (SONIX- of IFF-format) muziek- en desktopvideo- software gebruikt worden.

kende kogels en exploderend wapentuig. Naar keuze van links naar rechts of van rechts naar links. Ook musici zullen met deze optie wel raad weten om hun composities te versterken.
Vijf octafen staat gelijk met HIRES-graphics bij teken- en CAD- programma's. Net als bij de grafische beelden met een hoogoplossend vermogen vreet die betere kwaliteit vrij RAM- geheugen alsof het niets is. HiFi wordt nu eenmaal duur betaald. Een troost is dat Audiomaster II alle voor de Amiga gangbare geheugenuitbreidingen ondersteund. De echte muziekliefhebber zal vijf octaven in plaats van zeven (op de betere keyboards) wat magertjes vinden. Voor een computer in de Amiga-klasse is vijf octaven echter een hele prestatie.
Interne resampling. Andere geluids- en videosoftware kan vaak niet met hoge samplingrates uit de voeten. Met Audiomaster II kunt u in de Amiga tot een wel compatible aantal sps hersamplen. Dit gaat uiteraard wel gepaard met enig kwaliteitsverlies.
Het **aanpassen van de pitch**. Bijvoorbeeld een geluidsmonster aanpassen aan de toonsoort waarin de muziek waarmee het effect gemixed wordt gespeeld is.
Wat de **aanpassing** en **korrektie** van het gedigitaliseerde geluid betreft beschikt AudioMaster II over:

- **Auto peek adjustment** (het zelf uitregelen van geluidspieken) en **pre-scanning**. Beiden voor het reduceren van de vervorming.
- Een **ruisfilter** en een **aliasing filter**. Het eerste filtereffect dient als de treblecontrole, het tweede als een onderdrukker van de vervorming.
- Het **omzetten naar stereo**
- **Volumeregeling**

En verder biedt Audiomaster de nodige real-time special effects en mixages zoals:

- **Echo**
- **Reverse sampling**. Het achterstevoren opnemen/noteren van het geluidsmonster of -golf.
- Het **mixen van geluidsgolven** tot bijvoorbeeld een geheel nieuw electronisch muziekinstrument

Alle **editing procedures** gebeuren comfortabel op het monitorscherm met de Amiga-muis. CUT, PASTE, INSERT, REPLACE, COPY en het automatisch zetten van de REPEATS zijn onder volledige visuele controle (de geluidsgolven staan op de buis) eenvoudig uit te voeren. Dit editvenster functioneert tevens als oscilloscoop ter beoordeling van het inkomend geluid.
Tot slot nog de **RAM scan** waarmee de geheugeninhoud rechtstreeks naar de disk gedumpt kan worden.

De installatie

Audiomaster II stelt geen hoge hardware-eisen. Voor gewoon hobbygebruik zijn 512 KB aan vrij RAM en twee diskdrives zijn in principe voldoende. Voor echt professioneel gebruik zijn 1,5 tot 2 MB en een harde schijf een minimum-vereiste. Een

68020- versnellerkaart bespoedigt het rekenwerk.
Na het maken van een veiligheidskopie en het opstarten van het Audiomaster II-ikoon verschijnt een Aegis-requester met het versienummer, copyright-informatie en de hoeveelheid vrij RAM. Daarna krijgt u het hoofdscherm met bovenin de menubalk, in het midden het Edit-venster met de golfvormen en in het onderste schermkwart het controlepaneel. Het controlepaneel biedt voldoende mogelijkheden om eerst eens met de gedigitaliseerde geluiden op de datadisk te experimenteren. Heeft de gebruiker op deze manier wat ervaring opgedaan, dan kan het eigenlijke samplen beginnen.
Onder **Set Sampler Config** van het **Options Menu** staat, u raadt het al, hoe Audiomaster II op geleide van de sampler hardware in te stellen. Audiomaster is in vrijwel alle gevallen superieur aan de bij de sound sampler meegeleverde software, dus die kan de prullebak in. Klik in op de **Parallel**box en de samplesnelheid voor de Amiga CPU en de sampler kunnen worden ingesteld. Standaard zijn 22.372 voor stereo- en 44.744 voor monogeluid. Alleen met de 68020 microprocessor en een geschikte soundsampler is stereo/mono-sampling van 55.930 mogelijk! Bij het gebruik van alternatieve samplers bleek enig experimenteerwerk noodzakelijk. Na wat proefjes rolt de juiste sampling rate er wel uit.
Verder kan de gebruiker de volgende opties instellen:

- De voice activation **VOX**. Alleen geluid dat boven de ingestelde drempelwaarde uitkomt, wordt gesampled.
- Automatisch **bijregelen van de BIAS**. Alleen waar nodig op het scherm centreren. Hetzelfde geldt voor de automatische BIAS-berekening.

Na het saven van de instelinformatie onder Save Configuration behoeft de gebruiker voor het samplen alleen nog de **samplegrootte** in bytes en de **tijdsduur** in seconden op te geven. De formule:
Rate (sps) x time (sec)=size (bytes)
ligt daarmee vast.
Het samplen is verder een kwestie van de sampler hardware op de audiobron aansluiten en de optie **Sampler** uit het Projectmenu te activeren. De Digital Sampler-requster verschijnt in beeld en u kunt het inkomende volume onder Monitor en de samplegrootte regelen.



Schermbeeld van Audiomaster II

Het controlepaneel

Net als in een echte geluidsstudio biedt Audiomaster II een groot aantal controleknoppen. Alleen staan die knoppen nu op de monitor en worden zij via de muis bediend. De belangrijkste functies zijn:

- Het aanbrengen van **positie- en tijdcoördinaten**
- De **Display Size**. Het aantal samples op de twee stereokanalen
- Het aangeven van het werkgebied, **section** of **range**, voor het gebruik van de editfuncties op de geluidsgolf
- **In/uitzoomen**
- Het aanbrengen van **Repeat-markers**
- Het bepalen van lussen, **loops**
- **Snaphot** (tijdelijke opslag) en **Recall** brengt het snapshot weer terug in het editing-venster
- Het **afspelen** van het geluid

Kortom, een complete geluidstafel op de monitor.

Bewerken en saven

Het Edit-menu biedt de opties COPY (kopieert de data naar de kopieerbuffer), PASTE (zet de data uit de buffer op de door de muiscursor gemarkeerde plaats), CLEAR BUFFER (leegt de buffer), REPLACE (vervangt de door de cursor aangegeven data door de bufferinhoud), ZERO (voor het aanbrengen van werkruimte in de golf) en EDIT FREEHAND (voor het met de vrije muishand veranderen van de geluidsgolf). Onder het Special Effects-menu staande opties ECHO, BACKWARDS, MIX WAVEFORMS, CHANGE VOLUME, RESAMPLE, TUNE WAVEFORM, real-time DELAY en UNSAMPLE.
Het opslaan van het gedigitaliseerde geluid kan in normaal of **vijfoctaaf HiFi-format**. let daarbij op de juiste pitch, samplerate en de plaats van de herhalingspunten (Repeat Points). Gebruik bij voorkeur REPEAT ZERO voor het plaatsen van de markeringspunten. Gebruik de SNAPSHOT-functie voor het tijdelijk maken van backups voor het geval dat er bij de soundprocessing iets mis gaat!

Audiomaster II is geen allemans-software. Professionele sound sampling zal de gewone Amiga-hobbyist niet echt in vuur en vlam zetten. De nieuwigheid is er al gauw van af. Spelletjesmakers, desktopvideors, slideshowproducenten en electronisch georiënteerde musici zullen het gebodene zeker kunnen waarderen. De Engelse gebruiksaanwijzing is prima leesbaar en staat vol tips en vragen. AM II kost ƒ 279,-

Voor informatie: Altycos Imports, Zoetermeer: tel.: 079-510757.

U.S



Charlie Chip's
SOFT WIR WAR
DOOR: Tijs Meijer
...ZAL DE ELECTRICITEIT WORDEN UITGESCHAKELD!!
CHARLIE!!. ZE ROEPEN OM DAT DE STROOM ZO WORDT AFGESLOTEN!
WAT?!..
M'N COMPUTER!!
BAD
..ZAL IK 'M EVEN UITZETTEN DAN?!!
NEE!! IK MOET NOG IETS BEWAREN!
..MAAR IK GOOI OOK NIKS WEG!!
NEE.. AFBLIJVEN!
KOM JE UIT BAD DAN?!
NEE.. MAG NIET VAN DE REDAKTIE ER ZIT AL TEVEEL BLOOT IN DEZE PAGINA!
NOU.. DAN BEWAAR IK HET WEL EVEN!
ZEG MAAR WAT IK MOET DOEN!...
IN DE LA LINKS LIGT EEN LEGE WITTE DISK!!
ZE LIJKEN ME ALLEMAAL LEEG... EN ZE ZIJN ALLEMAAL ZWART!..
NOU.. PAK ER DAN MAAR EEN...'T MAAKT NIET UIT.. MAAR SCHIET 'N BEETJE...
NIEUW! NU MET MATTHEU
PLOP!
...OP....
EVEN LATER..
..WAT ROMANTISCH HE SCHAT... ZO'N AVONDJE ZONDER COMPUTER...
!